Hybrid Strategy of Particle Swarm Optimization and Firefly Algorithm for Black-box Function Optimization

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2021

Heng XIAO

LIST OF PUBLICATIONS

Journal paper

1. Heng Xiao, Toshiharu Hatanaka (in press). Model selecting PSO-FA hybrid for Complex Function Optimization. *International Journal of Swarm Intelligence Research*, 12, 4 (2021).

2. Heng Xiao, Gen Yokoya, Toshiharu Hatanaka (in press). Multifactorial Particle Swarm Optimization Enhanced by Hybridization with Firefly Algorithm. *International Journal of Swarm Intelligence Research*, 13, 1 (2022).

Conference paper

1. Heng Xiao, Toshiharu Hatanaka. Heterogeneous Swarm with First and Second Order Dynamics for Function Optimization. In *Proceedings of the 2016 IEEE Congress on Evolutionary Computation*, Vancouver, pp. 2077-2082, 2016.

2. Heng Xiao, Masato Oi, and Toshiharu Hatanaka. Comparison between PSO and FA about Agent's Moving Direction. In *Proceedings of the 10th SICE Symposium on Computational Intelligence*, pp. 35-38, December 16-17, 2016, Toyama.

3. Heng Xiao, Toshiharu Hatanaka. Hybrid Swarm with Property Changing Particles for Function Optimization. In *Proceedings of the 7th International Symposium on Computational Intelligence and Industrial Applications*, Beijing, FA-OS-04, pp. 1-4, 2016.

4. Satoru Iwasaki, Heng Xiao, Toshiharu Hatanaka, and Takeshi Uchitane. A general swarm intelligence model for continuous function optimization. In *Proceedings of the 11th International Conference on Simulated Evolution and Learning*, paper id. 218, November 10-13, 2017, Shenzhen, China.

5. Heng Xiao, Gen Yokoya, Toshiharu Hatanaka. Multifactorial PSO-FA Hybrid Algorithm for Multiple Car Design Benchmark. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Bari, October, 6-9, 2019.

Contents

Chapter 1

Introduction

People are always facing the problem of choosing. From all possible selections to find out the best one is not easy sometimes. The optimization problem is the problem that needs to find the best solution from feasible solutions. Conventional optimization problems are usually described as minimization problems. By computing and comparing the fitness of the solutions in one optimization problem, it is able to select at least one best solution out. However, some of the problems can be complex for processing. Among these problems, the black-box optimization problem is difficult since the landscape of the objective function is unknown.

Evolutionary computation [8] is inspired by biological evolution and provides many algorithms for optimization problems. Among the evolutionary computation, metaheuristics are provided by sampling a set of solutions. Metaheuristics utilize the search process to find near-optimal solutions in a search space. These algorithms are approximate and not problem-specific. Even though metaheuristics do not ensure to get the global optimum solution for some problems, they may get sufficiently good solutions for problems with incomplete information like the black-box problems. Some population-based metaheuristics called swarm intelligence (SI) [4, 29, 36] are inspired by the collective behavior of the natural or artificial dynamics biology systems. Individuals in a group can do personal work and take part in social work. They follow some common rules for cooperation.

When handling some problems that are modeled as mathematical functions, swarm intelligence simulates the cooperation mechanism of the natural group to build a population-based solver for them. By studying from the natural group work and utilizing the knowledge to develop tools, it is able to learn from the animals' behavior and summarize the rules of the behaviors. Then simulating the actual biology systems helps to build useful solvers. For optimization problems, both personal knowledge and social information are used for formulating the models of swarm intelligence.

Inspired by swarm intelligence, some optimization algorithms are developed for processing optimization problems. For each of them, some no mass agents are used to automatically search in a search space for finding candidate solutions. These agents' movements are affected by different factors. The agent group study from the surrounding situation when searching around a specific space. It means that each agent has a study ability to adapt itself to the environment. Besides, one would not only study about itself but also study the common knowledge which is shared by one common group. By using this method, the agents can cooperate with each other to do common work. Some interaction methods are provided in these swarm intelligence algorithms to help each agent exchanges its information with others. There are some well-known swarm intelligence algorithms such as particle swarm optimization (PSO) [30], firefly algorithm (FA) [61], brain storm optimization (BSO) [42], cuckoo search [9], bat algorithm [63], ant algorithm [43], artificial bee colony (ABC) [59] and so on. These algorithms have been used to process many optimization problems and have shown good performance on some of the problems. Improving methods for these algorithms have also been studied in these years.

As one of the population-based metaheuristic methods, PSO is easy to be implemented for optimization problems and has a strong global search. However, there are some reasons as the poor local search ability or premature convergence that cause PSO cannot keep finding better solutions in the latter evolution. Past

works provided ways to improve the conventional PSO as Gaussian mutation, random walk, position initializing, population reassignment, sampling (SPSO-2011). SPSO [64] proposes an idea of rotational invariance to enlarge the search range. A linear decrease inertia weight is used for building a balance of exploration and exploitation [58]. An approach called TVACPSO has been presented in which local best solutions are updated by using some mutation strategies as Cauchy, Gaussian, and opposition-based mutations [37].

However, a lot of them have the problem that delaying convergence and increasing the number of parameters that need to be adjusted. On another side, FA owns a strong local search ability for the optimization problems. Compared to simply importing the random walk, in FA, one firefly estimates the gradient of its own position by using the information of other points and search with a random walk biased towards the gradient [54]. It is important to make a good balance between the local search and the global search. Then consider how to make a better algorithm as a PSO-FA hybrid where different properties of PSO and FA are utilized.

PSO and FA are different about memory record and random factors. Making hybridization provides an inspiration to utilize distinct algorithms' properties. This provides a method that is possible to get better candidate solutions. By building a hybrid algorithm of PSO and FA, it is hopeful to improve search performance than PSO with the help of FA. This research focuses on designing an improvement strategy for the swarm intelligence model to utilize the properties of PSO and FA. With an appropriate strategy, the hybrid one is expected to improve the performance when handling some complex optimization problems.

Related works of making hybrid of PSO and FA are:

Modifying the standard model as: Let the attractiveness factor to instead the social and cognitive acceleration coefficients [32] and applied to network classifier. Let firefly owns the personal best of PSO [25] and applied to parameter selection in machining. Some other algorithms that the population changes from

one model to another model [2, 14]. Let firefly owns personal best and global best [38]. Changing particles' properties with a fixed rule: Changing particles to fireflies applied to power system [6], floorplanning problem [44]. Change the best firefly to particle [3] and applied to data classification. Changing the best fitness owner in the particles to firefly [48] and applied to the power system. Best particle moves as a firefly [34] and applied to satellite image classification. Making a sub-population of PSO and FA: Making sub-populations by sharing common optimal as FAPSO [50]. In FAPSO, a local search strategy is introduced to improve the exploitative capability. A multi-swarm method based on FA and PSO is proposed as a hybrid algorithm [27]. Some other hybrids based on sub-population are applied to cloud service [20], power management [33], tuning the penalized support vector machine's parameters [5]. Building a strategy to select the model of PSO or FA: HFPSO is proposed with a model selection strategy based on whether one particle improves than the previous global best record. Some works applied it to flow shop scheduling problem [28], forecasting solar power [1], electric power networks [40].

The above works using parameter tuning, modifying model and show good performance for specific problems. However, parameter tuning costs a lot of computation and it is not helpful to build widely-used solvers.

On another side, high-dimensional problems own many parameters and need many iterations or arithmetic operations for evaluations. In real applications, evaluating high-dimensional problems usually costs a lot of computation while large number evaluations may not be practical and reasonable. It is difficult to do parameter tuning during the whole evolution process. Besides, the black-box optimization problem [26] is difficult for the landscape of the objective function is unknown. It is challenging to get feasible and practical solutions for high dimensional black-box function optimization problems. Optimization solvers as the swarm intelligence that do not rely on problem-based knowledge are suitable to handle the black-box optimization problems.

For black-box functions, the analytic form is not known. A black-box function can be evaluated to obtain fitness values and approximated gradients. However, when evaluating a solution's quality for a metaheuristic, it needs to know the optimal solution in advance. For assessing the performance of optimization algorithms, the benchmark problems as the CEC benchmark are provided with known properties for testing to understand the weaknesses and strengths of one algorithm.

In chapter 3, a simple hybrid of PSO and FA is introduced. By testing the hybrid algorithm on CEC benchmark problems, it shows how controlling the composition of the hybrid group to affect the algorithm's performance. Different strategies are used to build hybrid mechanisms. For automatically changing the composition of the population, this research also proposed a model selection mechanism as an event-driven strategy that focuses on the personal study process. To show how the proposed strategy work, the proposed method is tested on some optimization problems and compared with some other swarm intelligence algorithms.

Then in chapter 4, a hybrid algorithm that uses a model selection mechanism is introduced. This research proposes making a hybrid swarm of standard models without parameter tuning. By utilizing an event-driven strategy that focuses on personal study to cutting off the effect of the global best for part of the population to let them find more potential solutions to improve the global best. HFPSO is a hybrid that utilizes a model selecting method based on whether a particle is successful to improve than the previous global best record. Compare to HFPSO, the proposed method focuses on the personal study process of the particles to improve the numbers of particles that update their personal best. Some expensive benchmark problems whose properties are known are provided. By testing the proposed hybrid on these problems to show how the proposed hybrid well performs on high dimensional problems than some other optimization methods. This research shows that the personal study is important to improve the performance of the whole group.

Besides, the PSO-FA hybrid swarm is applied to evolutionary multitasking [11, 24, 45, 67]. Evolutionary multitasking aims to simultaneously handle multiple optimization tasks when considering there is some relationship that exists among these tasks. As an evolutionary multifactorial optimization method, multifactorial optimization is proposed for handling multitasking problems with a single population. By updating a skill factor, one particle has a chance to change its preferred task. If one particle stop evolution when optimizing its current preferred task, it may lose to others that prefer the same task of it. Then changing its preferred task would give it a chance to improve its status in another task. Especially when the tasks are not similar to each other, one performs not well in a task does not mean that would be the same in another task for it. Then for one particle who fails to improve its best record in one task but success in another task, exploiting the current interested space is meaningful to contribute to the evolution in the reassigned task. The proposed method helps the particle to do the local search if it updates its personal best. Combining the proposed strategy with the multifactorial optimization to help the population to improve the solutions when processing multiple tasks. In chapter 5, the multifactorial hybrid algorithm is tested on a multifactorial benchmark problem and compared with a simple multifactorial particle swarm optimization to show that the combination of hybridization strategy and multifactorial optimization contributes to improving the performance of simultaneously handling multiple optimization tasks. In chapter 6, a black-box benchmark problem in which it needs to optimize the mass of three different cars is provided. Test the proposed multifactorial hybrid to show the method well performs on a real problem.

Chapter 2

Swarm intelligence

In the natural environment, there are many collective behaviors as ant colonies, bird flocking. Swarm intelligence is inspired by these collective behaviors in biological systems. In a swarm intelligence system, a simple population consists of some no-mass search agents with a cooperation mechanism. Each agent interacts with another one to process a common problem rather than a single individual to do that.

In a common group, members follow some rules to decide their actions. For the individuals, they own some inherent properties. The biological properties decide one individual can do what actions. In one group, the individuals cooperate with each other. Different kinds of communication methods exist in nature. By interacting with each other in a common group, individuals show their social behavior.

Swarm intelligence is inspired by the personal activity and the social behavior of the animal groups. Assume that there are some no-mass search agents instead of the actual moving animals to simulate the moving process. By utilizing the search behavior of a swarm intelligence model, the search agent population is able to explore the solution space of one problem with position updating.

On another side, optimization problems have got a lot of attraction in the field of economic, engineering. A metaheuristic is one method that relies on algorith-

mic knowledge rather than problem knowledge so it is able to be used to handle some expensive problems as the black-box optimization problems in which the landscapes are unknown.

Swarm intelligence [29] is one of the popular metaheuristics for handling the black-box optimization problem. Many swarm intelligence algorithms are developed with different population-based strategies. The swarm intelligence algorithms have some common properties of utilizing interaction among search agents and some stochastic factors. One general property of swarm intelligence for effectively handling optimization problems is parameter tuning of the algorithm parameters for a specific problem. However, it is difficult to tune the parameters without trial and error for that costs a lot of computation. This chapter introduces two swarm intelligence algorithms: PSO and FA. These two algorithms are different of memory record and stochastic factors. Then consider utilizing the differences of them to develop a better optimization solver.

2.1 Particle swarm optimization

Particle swarm optimization (PSO) [30] has been proposed by Kennedy and Eberhart in 1995. PSO is a biologically inspired optimization method. As a population-based search algorithm, PSO has got a lot of attention and has been applied to handle optimization problems. PSO has the merit that it is easy to be implemented.

About the implementation of PSO, it assumes that there are some no-mass particles that are randomly assigned in a search space. Whole particles move in the common search space with their velocities. Each particle gets its evaluation value by computing the position information in the objective function.

In the population, each particle will compare its current position with its previous position. Then, the agent will memorize its best position. For computing the optimum of the problem, the evaluation values of whole individuals are compared

and ranked. Then the best solution is able to be found. By comparing the best solution and the memorized best solution, it is able to know whether the global best position is updated. Through a large number of iterations, particles can iteratively update their best positions and share the knowledge with each other. When the terminating condition is satisfied, the best recorded solution is got.

In PSO, previous information is used for updating the current status. For each individual, its personal best position is recorded for the cognition process. The personal best information means one individual can memorize its previous experience and utilize the memory to contribute to the progress of itself. This factor causes one particle's behavior be affected by its memory. Besides, the social common knowledge as the global best position is recorded and shared among individuals in a common group. The group-shared information is updated by the leader individual and contributes to the whole group's communication. Common knowledge plays a role in adjusting one particle's velocity. The personal work and the social work are combined to construct one individual's moving velocity in PSO. Besides, one inertia factor is usually utilized in one individual's progress as another previous information. As an automatic searching algorithm, PSO uses one individual's previous velocity as an inertia factor. In most situations, only a part of one individual's velocity is used for future movement.

Some important issues for particle movements are the moving velocity, personal best position, global best position, current position, and so on. One particle refers to its personal best position p_{best} and group's best position g_{best} for its position updates. That means one agent learns from the previous knowledge, and its best knowledge is also shared with other members of the society. Besides, the moving velocity is recorded for each individual as an inertial factor. Each agent's current best value p_{best} is renewed in every step. At the same time, the swarm's optimum g_{best} is also renewed with agents' communication. And the velocity of agents also change with time. The personal best is studied and memorized by the agent itself. In a biological society, animals working with each other. Through an individual's

Table 2.1: Pseudo code of PSO

Step1	Initial the particle's position
Step2	Initial the particle's best known position, velocity
Step3	Set suitable parameters
Step4	Loop
	While (generation limit is not meet)
	Renew the particle's velocity
	Renew the particle's position and best known position
	Renew global best position
	Renew optimum
Step5	End loop

personal behavior, it can learn from the natural environment and accumulate experiences. By attending a social activity, the common knowledge would help the individual to adapt itself to the environment better.

PSO's update equations were defined as Eq (2.1):

$$\begin{cases} v_i^{t+1} = wv_i^t + c_1 r_1 (p_{best}^t - x_i^t) + c_2 r_2 (g_{best}^t - x_i^t) \\ x_i^{t+1} = x_i^t + v_i^{t+1} \end{cases} \tag{2.1}$$

Here, w is an inertia weight, c_1, c_2 are constants, r_1, r_2 are randomly selected from the uniform distribution over [0,1]. Indicates he velocity of particle i in time t, then the velocity v_i^t is updated to v_i^{t+1} in time $t+1$. With the velocity to update the position x_i^t to x_i^{t+1}. Then p_{best} indicates the personal best position, g_{best} is the global best position.

2.2 Firefly algorithm

Fireflies' flashing behavior is a well-known natural phenomenon. The flashing light of the firefly comes from the process of bioluminescence and helps to attract prey or protect the firefly itself from attack.

Fireflies communicate with each other by flashing. The brightness indicates the strength of the signal, then one firefly would be attracted by other fireflies

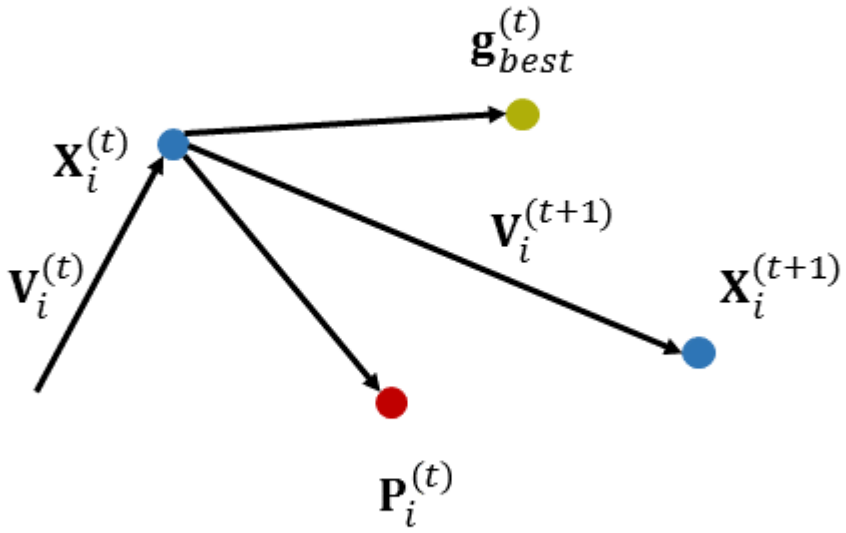


Figure 2.1: PSO model

that are brighter than it. By comparing the intensity, a firefly selects its flight direction. By simulating the flashing behavior and communication method of the firefly, a dynamics model for mathematical optimization is provided by X. S. Yang that called the firefly algorithm. The firefly algorithm has been used to process some optimization problems.

In the firefly algorithm, fireflies are flashing and moving at the same time. As an initial step, agents are randomly placed in the search space. Their intensities will be initialized with evaluation values of position information on the objective function. By ranking the intensities, one firefly is attracted by another one which is brighter than it. Define the firefly j that is brighter than i belongs to a common group G_i. The brightest one will move as a brown movement since it would not be attracted by other fireflies. Then agents move through the position's updating based on the communication among themselves. As a first-order dynamics model, there is no velocity issue used in the FA. Compare their intensity then move the agent which is less bright to the brighter one. It should be considered that the brightness will be changed with the distance r_{ij}. So attractiveness would be varied base on the Euclidean distance between agent i and j.

In the simplified model, it is assumed that a firefly selects its flight direction through the observed intensities in the flashing light. Light intensity at r distance is described as $I = I_0/r^2$ (I_0 is the intensity of the light source). For each firefly, light is absorbed and affected by a constant light absorption coefficient.

Firefly algorithm is developed by the flashing behavior of fireflies which is a familiar phenomenon. In the simplified model, it is assumed that the firefly selects its flight direction through the observed intensities in the flashing light. Then, FA's update equation was proposed as Eq (2.2).

$$x_i^{t+1} = x_i^t + \sum_{j \in G_i} (\beta \exp[-\gamma r_{ij}^2](x_j^t - x_i^t) + \alpha \varepsilon_t) \tag{2.2}$$

Here, β indicates an attraction weight to the brighter fireflies, and γ represents

Table 2.2: Pseudo code of FA

Step1	Initial the fireflies's position
Step2	Initial intensities with function fitness
Step3	Set suitable parameters
Step4	Loop
	While (generation limit is not meet)
	Renew the position and evaluate
	Ranking intensities of fireflies
	Find best firefly
	Renew optimum
Step5	End loop

the light absorption rate according to the distance. For each individual i, one individual is brighter than i is indicated as j. r_{ij} means the distance between i and j. Then, in the last term, α is a parameter for controlling the step size, and ε_t is a random vector for a random walk whose elements are sampled from Gaussian distribution with 0 mean and unit variance. Then update the position x_i^t to x_i^{t+1}.

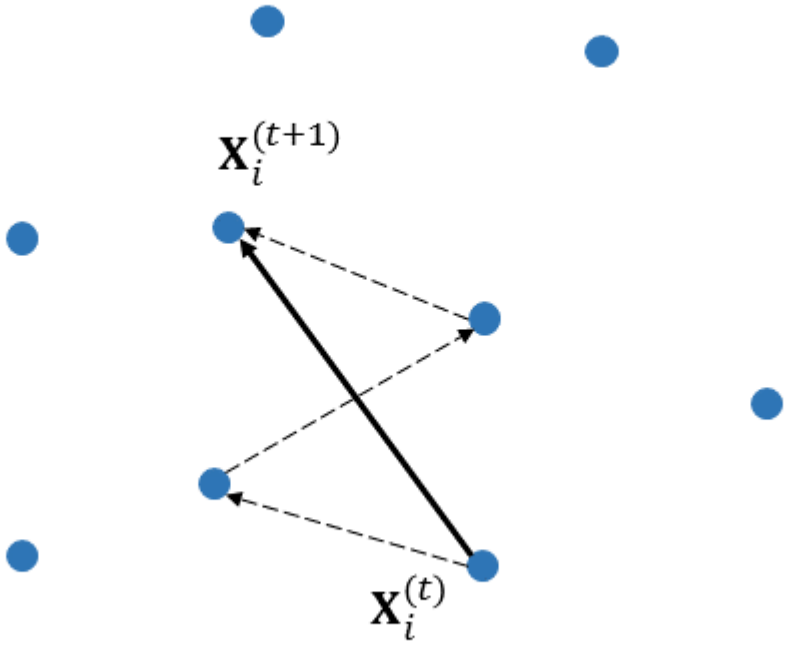


Figure 2.2: FA model

Chapter 3

Hybrid algorithm for black-box optimization problem

Optimization is a rational requirement in various fields as engineering designing, economic planning, and so on. There are various kinds of optimization problems that aim to obtain good solutions. Among these optimization problems, a class of black-box function optimization [26] gets the attraction of researchers. Black-box means that only the output value is available at the observed points of design variables. Also, any information about a derivative and whole landscape of the objective function is not available in computing optimum solutions. Thus, an automatic search algorithm is employed to deal with such kinds of problems. In particular, a metaheuristic [62] approach has been an attractive method since it requires algorithmic knowledge rather than a heuristic with domain-specific knowledge. However, generally speaking, while facing different problems, metaheuristics do not always perform better than a heuristic one. It is important to make a metaheuristic search method work well in different situations.

Metaheuristic methods can be categorized as local search or global search, single-solution based or population-based, and so on. Since a single model can well solve only parts of the optimization problems, we may require to think about other ways of paralleling solving problems better. To this end, there have been

proposed plenty of efficient algorithms with their improvements in dealing with complex optimization problems. For example, making a hybrid algorithm may be a good way. However, some of such hybrid methods were ad-hoc approaches. Even if such approaches achieved good performance, they were sometimes criticized. So we should employ more interpretive approaches. By making use of the properties of base algorithms, it will be possible for us to build a well-performed combination.

This chapter introduces a simple hybrid algorithm that is combined with particle swarm optimization and firefly algorithm. Then it will introduce how to set the population with different models and describe how they sharing common information with each other. Then, the algorithm is tested on 15 complex functions that were supplied by CEC 2015 single-objective computationally expensive optimization problems [10]. From the computing results, it expects to find some influencing factors of the hybrid algorithm and consider how to improve the algorithm performance.

3.1 Simple hybrid of PSO-FA

In this section, a simple hybrid algorithm for PSO and FA will be explained. In a common group for hybridization, two different kinds of individuals exist. For developing a PSO-FA hybrid algorithm for optimization problems, a hybrid mechanism that fixing the numbers of particles and fireflies is proposed.

In PSO, the particles share the global best information as common knowledge. While in FA, the fireflies compare with each other by ranking the intensities of them. For combining the two different types of agents together, it is necessary to let all agents share some information in common. In the proposed hybrid algorithm, all agents have their own intensities no matter they are particles or fireflies. At the same time, whole individuals would share the global best memory. Particles' number and fireflies' number would be fixed in the hybrid swarm. It means that the percentage of the different types of individuals can be consciously

Table 3.1: Pseudo code of PSO-FA hybrid algorithm

Step1	Set numbers of particle (n_1), firefly (n_2)
	Initialize the agents' position
Step2	Evaluate the agents and initial the global best and personal best
Step3	Loop start
	If the agent is a particle
	move as particle according to Eq. 2.1
	Elif the agent is the firefly
	move as firefly according to Eq. 2.2
Step4	Evaluate the agents
	Update the global best and personal best
Step5	If the terminate condition is satisfied
	End loop
	Else
	Go to step 3

tuned.

The working flow of the hybrid swarm is simple. As an initial step, the individuals will be randomly placed in the search space. Then evaluating each one by calculating the position in the objective function. Set their initial positions as personal best position and find the global best position by comparing the evaluation values of them. In each iteration, some individuals would move according to the PSO model, while others would move according to the FA model. At the same time, the global best memory would be shared by the whole society and used for particles' study process. Besides, all agents own the intensity issue and the intensity will be initialized with the function value. In each iteration, whole agents contribute to updating the global best memory. Fireflies would be affected by both particles and fireflies. Within the limited computation resource, the final optimum is got. The hybrid algorithm is shown in Table 3.1.

3.2 Benchmark optimization problem

For one minimum optimization problem, there is a definition space in which it is able to find solutions. Each of them is represented by a vector with multiple variables. By searching in the space, some candidate solutions can be found for comparison. The objective of the optimization solver is find the best candidate solution from whole solutions within the defined space. By changing the design variables of one vector, it is able to get different solutions. Then evaluate the candidate solutions according to calculate the fitness of them in the objective function. By computing and comparing corresponding evaluation values, the best solution can be selected out from whole possible solutions. By iteratively evaluating, updating, and comparing the solutions, one algorithm is able to find at least one best solution within limited computation resources.

Black-box optimization is provided as the objective function is unknown. For developing efficient algorithms, it cost a lot of evaluation time of objective functions in real black-box optimization problems. On another side, some benchmark problems are provided with expensive settings that limited computation is affordable. The test suites as the CEC benchmark are provided with limited computation times. These benchmarks are important to assess optimizers' performance, understand one algorithm's weaknesses and strengths, and contributes to testing new algorithms. Then in this research, the benchmark test suites are used for developing algorithms for that these problems with given functions can be simply calculated. In addition, instead of costing too much computation resource on the trial and error about parameters, making a good composition of models to improve the search performance that finding better solutions is important for the application of swarm intelligence.

In CEC 2015, some bound-constrained single-objective computationally expensive numerical optimization problems are provided for testing algorithms' performance. The provided problems are quite more difficult than some common

optimization problems. Whole problems are functions with 10 or 30 dimensions. Some of the problems are shifted or rotated to improve the difficulty of processing. They are unimodal, multimodal and some of them are called hybrid functions composited of several basic functions.

The basic functions for the CEC 2015 expensive single-objective optimization problems [10] are described as below.

1. Bent Cigar Function

$$f_1(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2 \tag{3.1}$$

2. Discus Function

$$f_2(x) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2 \tag{3.2}$$

3. Weierstrass Function

$$f_3(x) = \sum_{i=1}^{D} \left(\sum_{i=0}^{kmax} \left[a^k \cos\left(2\pi b^k (x_i + 0.5)\right) \right] \right) - D \sum_{i=0}^{kmax} \left[a^k \cos(2\pi b^k \cdot 0.5) \right] \tag{3.3}$$

a=0.5, b=3, kmax=20

4. Modified Schwefel's Function

$$f_4(x) = 418.9829 \times D - D \sum_{i=1} g(z_i) \qquad z_i = x_i + 4.209687462275036e+002$$

$$g(z_i) = \begin{cases} z_i \sin(\mid z_i \mid^{\frac{1}{2}}) \quad if \mid z_i \mid \leq 500 \\ \Big(500 - \bmod(z_i, 500)\Big) \sin\Big(\sqrt{\mid 500 - \bmod(z_i, 500) \mid}\Big) \\ -\dfrac{(z_i - 500)^2}{1000D} \quad if\ z_i > 500 \\ \Big(\bmod(\mid z_i \mid, 500) - 500\Big) \sin\Big(\sqrt{\mid \bmod(\mid z_i \mid, 500) - 500 \mid}\Big) \\ -\dfrac{(z_i + 500)^2}{1000D} \quad if\ z_i < -500 \end{cases} \tag{3.4}$$

5. Katsuura Function

$$f_5(x) = \frac{10}{D^2} \prod_{i=1}^{D} (1 + i \sum_{j=1}^{32} \frac{\mid 2^j x_i - \text{round}(2^j x_i) \mid}{2^j})^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} \tag{3.5}$$

6. HappyCat Function

$$f_6(x) = \Big| \sum_{i=1}^{D} x_i^2 - D \Big|^{1/4} + (0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5 \tag{3.6}$$

7. HGBat Function

$$f_7(x) = \Big| (\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2) \Big|^{1/2} + \frac{0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i}{D} + 0.5 \tag{3.7}$$

8. Expanded Griewank's plus Rosenbrock's Function

$$\begin{aligned} f_8(x) =& f_{11}(f_{10}(x_1, x_2)) + f_{11}(f_{10}(x_2, x_3)) + \cdots + f_{11}(f_{10}(x_{D-1}, x_D)) \\ &+ f_{11}(f_{10}(x_D, x_1)) \end{aligned} \tag{3.8}$$

9. Expanded Scaffer's F6 Function

$$\begin{aligned} g(x, y) &= 0.5 + \frac{\sin^2 \sqrt{(x^2 + y^2)} - 0.5}{(1 + 0.001(x^2 + y^2))^2} \\ f_9(x) &= g(x_1, x_2) + g(x_2, x_3) + \cdots + g(x_{D-1}), x_D) + g(x_D, x_1) \end{aligned} \tag{3.9}$$

10. Rosenbrock's Function

$$f_{10}(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{3.10}$$

11. Griewank's Function

$$f_{11}(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{3.11}$$

12. Rastrigin's Function

$$f_{12}(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10) \tag{3.12}$$

13. High Conditioned Elliptic Function

$$f_{13}(x) = \sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}}x_i^2 \tag{3.13}$$

14. Ackley's Function

$$f_{14}(x) = -20\exp\left(-20\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e \tag{3.14}$$

and so on.

Then 15 expensive single-objective optimization problems are changed from these basic functions with some rotate and shift data. They are complicated and not easy to get solutions for them. The detail of the problems is shown in Table 3.2.

3.3 Numerical test

In this test, the PSO-FA hybrid algorithm is used to compute these complex optimization problems' minimums. The agent numbers are set as n_1 and n_2, which will moving according to PSO rules or FA rules. The test condition is

Table 3.2: CEC 15 expensive optimization test problems

Categories	F_i	Functions	F_i^*
Unimodal functions	F_1	Rotated Bent Cigar Function	100
	F_2	Rotated Discus Function	200
Simple Multimodal functions	F_3	Shifted and Rotated Weierstrass Function	300
	F_4	Shifted and Rotated Schwefel's Function	400
	F_5	Shifted and Rotated Katsuura Function	500
	F_6	Shifted and Rotated HappyCat Function	600
	F_7	Shifted and Rotated HGBat Function	700
	F_8	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	800
	F_9	Shifted and Rotated Expanded Scaffer's F6 funtions	900
Hybrid funtions	F_{10}	Hybrid Function 1 (N=3)	1000
	F_{11}	Hybrid Function 2 (N=4)	1100
	F_{12}	Hybrid Function 3 (N=5)	1200
Composition functions	F_{13}	Composition Function 1 (N=5)	1300
	F_{14}	Composition Function 2 (N=3)	1400
	F_{15}	Composition Function 3 (N=5)	1500

described as below:

Agent numbers: 50, n_1(particle number), n_2(firefly number); PSO parameter: c_1=1.4, c_2=1.4, w=0.7; Implement times: 30; Dimension: 10; Search space range: [-100, 100].

The 15 optimization functions were tested with 1000 iterations. Each agent will be initialized randomly in the search space before the iteration starts. In every iteration, a particle will update its local best memory and global best memory while the brightness firefly will also contribute to updating the global best memory. In addition, the test result will be expressed as taking the logarithm of the objective function value.

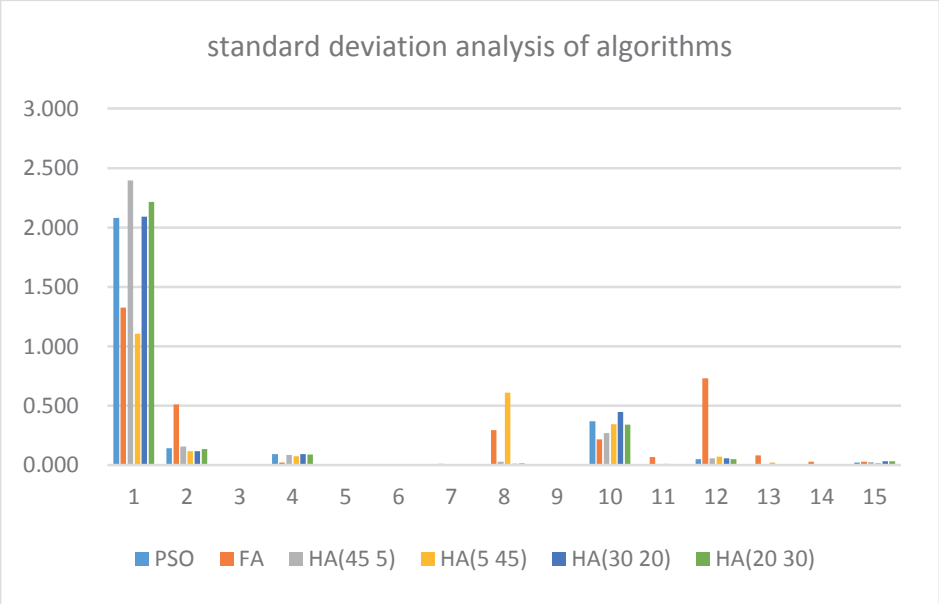


Figure 3.1: Standard deviation analysis of algorithms (x axis shows the functions' numbers, y axis shows the standard deviations of 30 runs

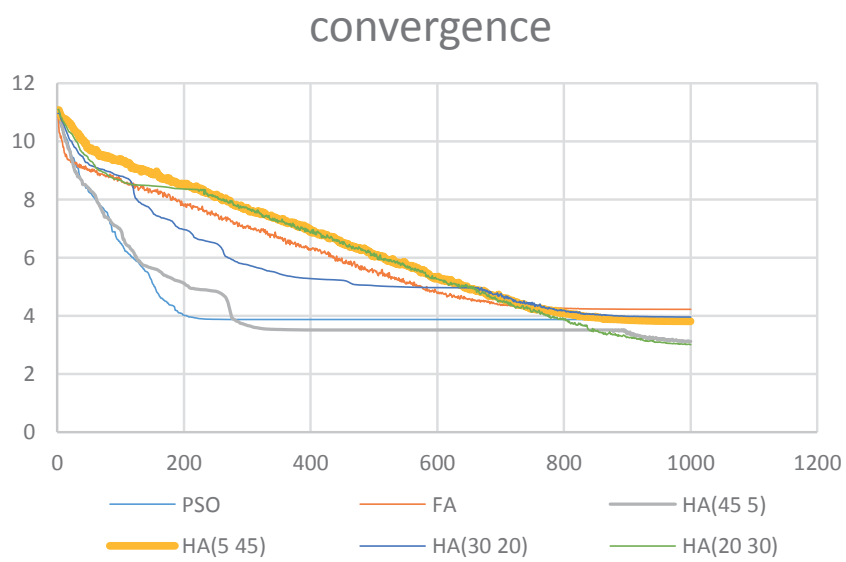


Figure 3.2: Convergence (x axis show the loops' numbers, y axis shows the logarithms of the global best values)

Table 3.3: CEC15 benchmark problem test result

		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
pso	min	2.214	3.227	2.477	2.726	2.699	2.778	2.845	2.903	2.955	3.135	3.042	3.088	3.208	3.200	3.177
	max	4.430	4.234	2.491	3.064	2.700	2.778	2.845	2.904	2.956	4.212	3.045	3.176	3.210	3.206	3.281
	std	0.491	0.267	0.003	0.100	0.000	0.000	0.000	0.000	0.000	0.260	0.001	0.020	0.000	0.001	0.049
fa	min	3.353	4.896	2.493	3.386	2.701	2.781	2.871	4.291	2.956	5.655	3.054	3.235	3.245	3.209	3.297
	max	10.296	7.867	2.499	3.539	2.707	2.786	2.939	6.131	2.957	8.051	3.146	3.360	3.437	3.236	3.362
	std	1.217	0.875	0.002	0.036	0.001	0.001	0.018	0.513	0.000	0.519	0.021	0.031	0.057	0.008	0.015
ha(45 5)	min	2.816	3.243	2.478	2.639	2.699	2.778	2.845	2.903	2.955	3.151	3.042	3.087	3.208	3.201	3.177
	max	4.430	4.229	2.491	3.082	2.700	2.778	2.845	2.904	2.956	3.773	3.045	3.157	3.210	3.206	3.298
	std	0.404	0.271	0.003	0.116	0.000	0.000	0.000	0.000	0.000	0.171	0.001	0.013	0.000	0.001	0.049
ha(5 45)	min	2.315	4.006	2.479	2.738	2.699	2.778	2.845	2.903	2.956	3.389	3.043	3.089	3.208	3.202	3.180
	max	7.197	4.972	2.493	3.263	2.701	2.779	2.846	2.938	2.956	5.823	3.061	3.175	3.216	3.208	3.325
	std	0.918	0.216	0.004	0.143	0.000	0.000	0.000	0.006	0.000	0.655	0.003	0.022	0.002	0.002	0.030
ha(30 20)	min	2.000	3.464	2.480	2.718	2.699	2.778	2.845	2.903	2.955	3.212	3.042	3.089	3.208	3.201	3.177
	max	4.430	4.387	2.489	3.090	2.700	2.779	2.846	2.904	2.956	4.003	3.045	3.168	3.214	3.208	3.295
	std	0.710	0.224	0.003	0.101	0.000	0.000	0.000	0.000	0.000	0.227	0.001	0.024	0.001	0.002	0.051
ha(20 30)	min	2.000	3.246	2.478	2.801	2.699	2.778	2.845	2.903	2.955	3.232	3.042	3.089	3.208	3.201	3.178
	max	4.430	4.487	2.492	3.216	2.700	2.778	2.846	2.905	2.956	4.469	3.049	3.176	3.250	3.206	3.295
	std	0.578	0.270	0.003	0.099	0.000	0.000	0.000	0.000	0.000	0.294	0.001	0.024	0.007	0.001	0.048

PSO, FA and the PSO-FA hybrid algorithm (4 types: particle number = 45, firefly number = 5; particle number = 5, firefly number = 45; particle number = 30, firefly number = 20; particle number = 20, firefly number = 30;) are ran under the 10 dimension situation. This computation is ran about 30 times for all 15 complex optimization problems. The implement result is shown in Table 3.3.

As we can see from the Table 3.3, some results such as the minimum value, the maximum value and the standard deviation of each function's test are recorded.

The PSO-FA hybrid algorithm deal with most of the problems well. From Table 3.2, it is able to see that the theoretical final optimum of these complex functions. These algorithms got optimums through lots of loops. By comparing the test result with each other we can see that these algorithms performed well on lots of functions.

We can see that these algorithms performed differently when they dealing with different functions. It seemed that HA(30 20) and HA(20 30) can get the best minimums of F_1. To other functions, HA(30 20) and HA(20 30) can also perform as well as PSO.

As we have seen in the Figure 3.1, there are the standard deviations of test results. From the graph, when testing the F_1 and F_{10} with PSO, FA and HA, these algorithms' test results performed big standard deviations. But to other functions, the test results are stable with small standard deviations. And to lots of functions, FA's results are more unstable than others.

In order to show the difference between these algorithms' evolution speed, here provides a graph of one run's convergence process of F_1. From the Figure 3.2, it is able to see that PSO gets good fitness fast while FA improves slowly. It shows that the hybrid algorithm performs between them. In the hybrid algorithms, if there are more particles than fireflies, the speed for finding better global best will be faster.

PSO and FA have been used for processing optimization problems by lots of researchers. By combining the properties of these two models, individuals with

different characteristics can cooperate with each other in a common group. By building and testing the PSO-FA hybrid algorithm in which the number different kinds of particles are fixed, it shows that tuning the number of particles can change the algorithm's performance.

Chapter 4

Model selecting PSO-FA hybrid algorithm

This chapter introduces a proposed method with model selecting strategy for a hybrid of PSO and FA [53]. PSO utilizes the previous information as personal best for personal study and global best for social communication. Since PSO has some drawbacks as poor local search or premature that would cause stop evolution, previous works provided a lot of methods to improve the performance of the PSO. For example, SPSO [64] proposes an idea of rotational invariance to enlarge the search range. A linear decrease inertia weight is used for building a balance of exploration and exploitation [58]. Besides, making a hybrid algorithm is one method to improve PSO by using the properties of another algorithm. Many PSO-related hybridizations have been developed in recent years. For each of these algorithms, a common population is expected to perform properties of different models.

The PSO-GA hybrid has been developed by involving the Genetic Algorithm (GA) and PSO as inputting the GA character into PSO. The selection, mutation, and reproduction of the population are used in the PSO-GA hybrid [39]. DE is also presented as one of the metaheuristics for processing global optimization problems. Using the property of DE is hopeful to improve population diversity.

By combining the Differential Evolution (DE) with PSO, DEPSO [65] is provided. Besides, the hybrid of PSO and other algorithms such as Simulated Annealing [60], Ant Colony Optimization [41], Cuckoo Search [19], Artificial Bee Colony [15] have been provided in recent years.

Different hybrid mechanisms are provided for building hybrid algorithms. Except of making a fixed percentage of different particles, there are other methods such as setting a switching condition for controlling different algorithms. One switching condition is that setting a number before which one algorithm works and then another one works. For the information communication among the hybrid population, there is a method in which the best particles of GA and PSO are exchanged [39]. In the past years, the PSO-GA hybrid is used to process many problems such as unconstrained global optimization problems and some engineering design optimization problems.

Making a good balance between exploitation and exploration is important for one swarm intelligence algorithm [18]. In a general swarm intelligence model, adjusting the parameters can change the effect of them [23]. For a hybrid swarm of different models, how controlling the exploitation and exploration would affect its performance. Then making a good balance between the PSO and the FA is important to develop a well performed hybrid swarm. PSO and FA have some common points about position update for that they all own the iteration mechanism in the swarm and stochastic factors. At the same time, they have some different properties: 1. PSO uses the previous knowledge for iteration information while FA has no memory and uses current information. 2. PSO uses a random factor to adjust the effect of the cognition and social learning while FA uses a random walk. Different from PSO, FA relies on the current information and the fireflies communicate with each other by comparing the intensities of themselves. This contributes to avoiding been affected by negative previous knowledge. Making a hybrid swarm of PSO and FA is expected to improve the performance of PSO about getting better global solutions.

Some hybridizations [7, 51, 52] of PSO and FA have been presented. Xiao and Hatanaka presented a simple hybridization with fixed numbers of particles and fireflies [51]. By tuning the proportion of two different types of agents, the performance of the hybrid swarm would be changed. It shows that tuning the population component would affect the whole group' performance. Finding a suitable component proportion would improve the performance of the hybrid swarm.

Besides the fixed component, there is a method in which one agent's character is able to be transformed. A property changing method is proposed aim at changing the characteristic of a specific individual of the hybrid swarm [52]. This method is applied to build a property changing PSO-FA hybrid algorithm. In this algorithm, the initial population is composed of some particles and fireflies. When the agent group iteratively searching for finding better solutions, the best particle would be transformed into a firefly. After the movement, the transformed one changes back to a particle. This algorithm shows that changing character can affect the whole group's performance.

With property changing mechanism, one individual is able to perform different characters. However, changing a specific individual may not be good for processing all problems. Automatically changing the character is a good method for helping one individual select a preferred model based on the actual situation. Different from the method of changing a specific individual, there is a method called HFPSO [7] in which one agent can be automatically transformed. In this method, one particle changes its preferred model based on the status of the evolutionary process. The individual implements the FA model when it improves than the previous global best, or it will implement the PSO model. In HFPSO, it uses a memetic method with the model selection to control agents' movement. By comparing the fitness of the agents' current position and fitness of the previous global best, it can judge whether it gets better fitness than the global best. HFPSO aims to control individuals' characteristics in global search stage or local search stage to improve the algorithm's performance.

HFPSO is a hybrid that utilizes a model selecting method with an event-driven trigger of whether a particle is successful to update the previous global best record. Different from HFPSO, this chapter proposes a hybrid strategy that focus on the personal study process. When one particle improves than its personal best record, then let it to move as a firefly. This strategy aims to show that the personal study utilizes FA to do local search helps to improve the performance of getting better solutions. CEC benchmark provides some expensive optimization problems. By testing the hybrid swarm on the benchmark problems to observe whether the hybrid swarm performs better of getting global best solutions.

4.1 Model selecting strategy

There are some reasons cause PSO cannot keep finding better solutions in the latter evolution. Strong local search ability can lead to the particles quickly gathering around the local optima and that would cause the premature convergence of the population. When the local search ability is weak, it will be difficult for the particles to convergence to a local optimum. These can cause the global best to stop improvement. When processing black-box optimization problems, it is difficult to observe the behavior of the particles. However, it is able to imagine that many particles fail to improve themselves when the global best stops improvement. Find a way to improve the number of particles that update their personal best is expected to improve the global best.

For one problem, particles are easy to find some interesting zones for themselves. Since particles are attracted by the global best, it may limit the population to find better solutions if the global best stop evolution and attract whole particles to leave away from their current interesting space. By changing the characteristic of one particle and make it move as a firefly, it helps the particle to get rid of the effect of the global best and focus on their interested space. With the local search, it increases the possibility for particles to find better solutions than the

current global best solution. On another side, HFPSO is a hybrid that utilizes a model selecting method based on whether a particle is successful to improve than the previous global best record. For each particle, it fails to improve than the previous global best does not mean that it fails to improve its personal best record. (When whole particles fail to improve than the global best, the hybrid swarm would just be changed to a PSO population) Only the outstanding particles would have a chance to do the FA movement for local search. Different from HFPSO, this research proposes a hybrid strategy that focuses on the personal study process. When one particle improves than its personal best record, then let it move as a firefly. It means that one particle would do the local search to exploit the current interested space until its evolution stops.

This section explains the hybrid swarm model with PSO and FA based on event-driven model change (HA). The hybrid algorithm is proposed by utilizing a model selection strategy based on whether the personal best is updated. If the personal best is updated, it would be handled by FA or it would be handled by PSO. The model changing aims to guide the agents that focus om their surroundings to do the local search when they succeed to improve their previous status or following the leader to do the global search when they fail to improve themselves. This method aims to enlarge the potential possibilities of finding better solutions.

For each individual i, it decides the next movement model by justifying whether the personal best is updated. Here $f(x_i^t)$ denotes the fitness of individual i, and $f(p_{best_i^t})$ denotes the fitness of its personal best. If the individual gets a better evaluation than the personal best at the current time t, then the personal best is updated. In the proposed model, this updating event is used as the trigger for model change. Thus, each individual would update its position according to the following model,

The definition of the model selection mechanism is shown in Eq 4.1:

Table 4.1: Pseudo code of PSO-FA hybrid algorithm

Step1	Initialize the individual position
Step2	Evaluate individuals and set the personal best p_{best_i}
Step3	If terminate condition not meet
	Find the global best g_{best} from $p_{best_i}, (i = 1, 2, \ldots, N)$
	For each individual i
	If the personal best is updated
	move as firefly according to 2.2
	Else
	The agent moves as a particle according to 2.1
	Evaluate the fitness
	If $f(x_i) < f(p_{best_i})$:
	Update the p_{best_i}
Step4	End

$$\text{Agent moves as} \begin{cases} \text{PSO, if } f(x_i^t) \leq f({p_{best_i}}^{(t-1)}) \\ \text{FA, if } f(x_i^t) > f({p_{best_i}}^{(t-1)}) \end{cases} \tag{4.1}$$

The PSO-FA hybrid algorithm is built as in Table 4.1. In the proposed algorithm, the model selection mechanism based on an event-driven mechanism of personal best information update. HA uses the model selection strategy of p_{best} to decide the moving model of the agent.

The algorithm is used for handling optimization problems. The detail optimization process starts with the initial population setting. Whole search agents in the population are randomly located in a search space. Each one's position is changed with the movement and described as x_i^t. By updating the individual position x_i^t of the i-th individual in discrete time t, it can search in a continuous space. For one which implements the PSO movement, it owns a velocity v_i^t. Through its personal study and social communication, it updates its velocity in the next time step for changing the position. For one which implements the FA movement, it uses an attractive factor to update the position. One would get a fitness number by applying its position vector to the objective function of the problem. By comparing and ranking whole particles' fitness, the best fitness is

found. If the best fitness improves the previous best fitness, then it is recorded for updating the best memory solution.

Assume that for one optimization problem that owns high dimensions, in local search stage of PSO, the fast convergence ability slows down when particles searching close to an optimal solution in the solution space. Then the particles would oscillate around optimal solution and cause delay of the optimization task. In FA, fireflies not own velocity and previous best position. It means that fireflies move regardless of their previous best positions and this may be useful in the exploitation stage for each particle. Then in the proposed strategy, by changing the moving method, the particle has the chance to move as a firefly. This may help the particle run away from the previous best position.

4.2 Numerical test

CEC2015 and CEC2017 provide some computationally expensive problems for performance comparison of algorithms. This section provides the numerical test of PSO, FA, SPSO, HFPSO, HA on these problems. Limited evaluation times are allowed for each algorithm. This paper provides the numerical test on the CEC15 benchmark problems and CEC17 benchmark problems [49] which is described in Table 4.2. The functions in both benchmarks are separated into four categories: unimodal, simple multimodal, hybrid and composition. These unimodal or multimodal functions are non-separable. Whole functions are provided in high dimensions and in each dimension the search space is limited to [-100,100].

The proposed method focuses on the personal study process of the particles. This method is expected to improve the numbers of particles that update their personal best by changing characteristics. Here show some examples of one run's result in Figure 4.1, 4.2, 4.3. The particle group totally owns 50 particles. For the change curve of the particles' number, the vertical axis means the number of the particles that succeed to update their personal best in one iteration. The

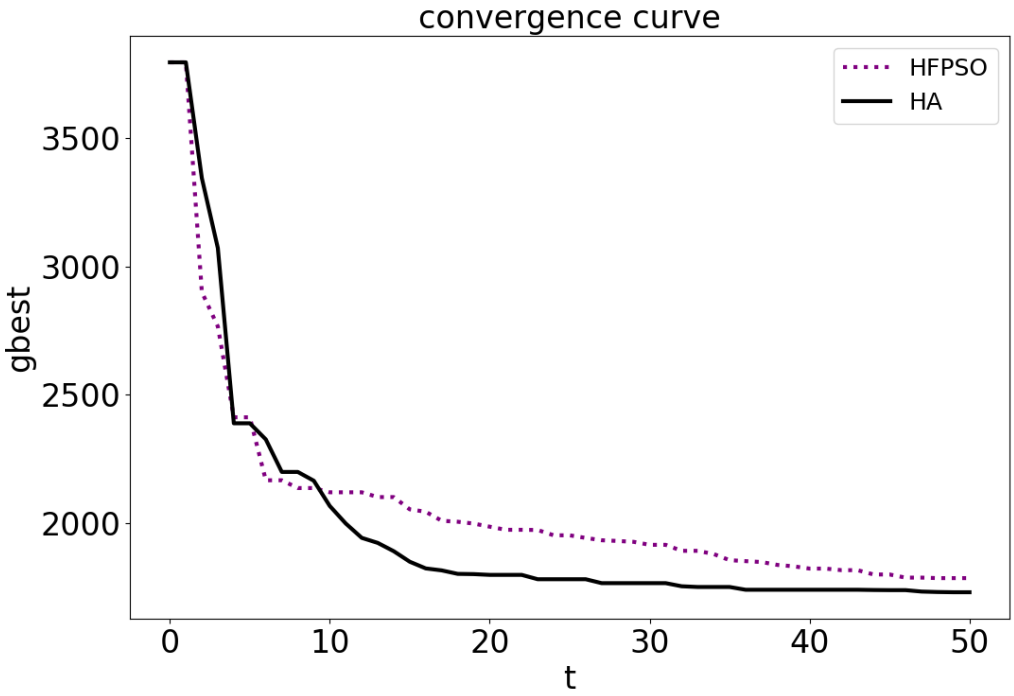
number in the vertical axis shows how many of them improve themselves. The figures describe the evolutionary process about personal best and global best of the proposed method and HFPSO in three functions: F_{13}, F_{14}, F_{15} of CEC 2015. From the test result, it can see that compare to HFPSO, the proposed method apparently improves the numbers of particles that update their personal best. And the global best is also improved than HFPSO. The proposed method is effective to improve the performance to find better solutions.

In the test, it compared the hybrid algorithm with simple PSO, SPSO (Bigiarini et al., 2013), FA and HFPSO. About the test environment, here describe the parameters as below: The total evaluation number is set as 1500 for 30 dimensions (30D). For the PSO model, c_1 and c_2 are set as 1.4. Then in this paper, the inertia weight w is set as 0.8. Randomness r_1, r_2 generated follow $U(0, 1)$. For the FA model, β=2.0, γ=1.0, and the step scale α use 5. The randomness of the random walk is generated by following $N(0, 1)$. The test will implement for 20 times for each case. The mean value of the 20 times run would be recorded for comparison.

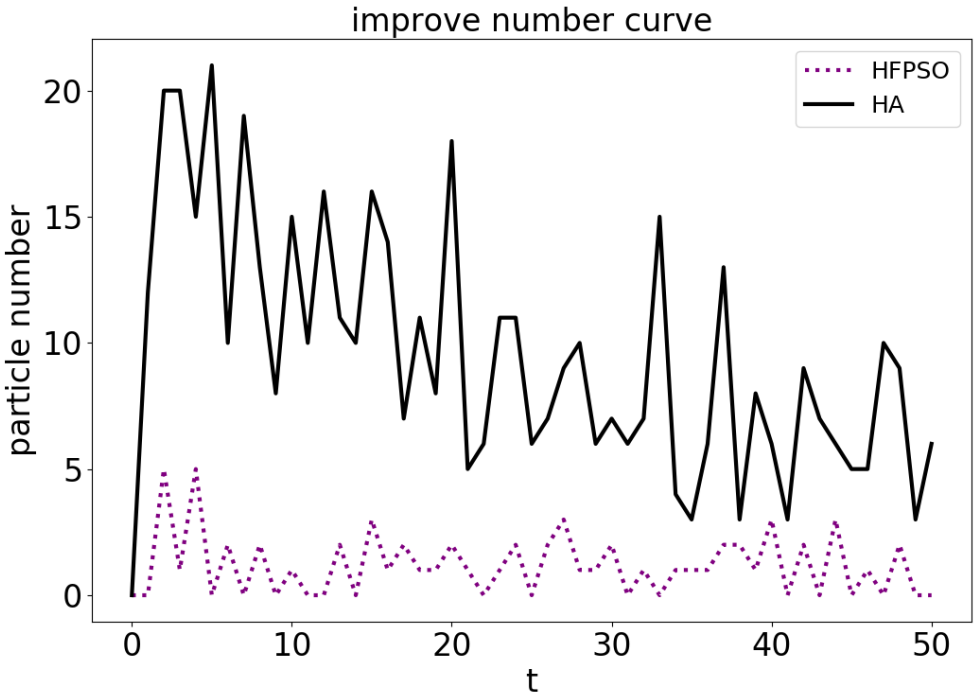
In Figure 4.4, the horizontal axis means the iteration numbers while the vertical axis means function values of the best memory record. Then use the logarithm value to scale the fitness to make the graphs. These figures describe the update situation of the global best information in each algorithm. In the result of F_1 - F_{15}, the hybrid algorithm HA performs better evolutionary trends than simple PSO and FA. Besides, the hybrid algorithms, get better optimums than other algorithms averagely. It means that the hybrid algorithm averagely performs better than PSO, FA, and SPSO.

The proposed PSO-FA hybrid algorithms is designed for developing a hybrid swarm that can well process more optimization problems than PSO or FA for optimization problems. A model selection strategy is used in the hybrid swarm.

From the result of the CEC 2015 benchmark, it is able to observe how these algorithms perform differently on the 15 problems. The result is shown in Table 4.3 and Table 4.5. For the F_1 function, it is unimodal with a smooth but narrow ridge

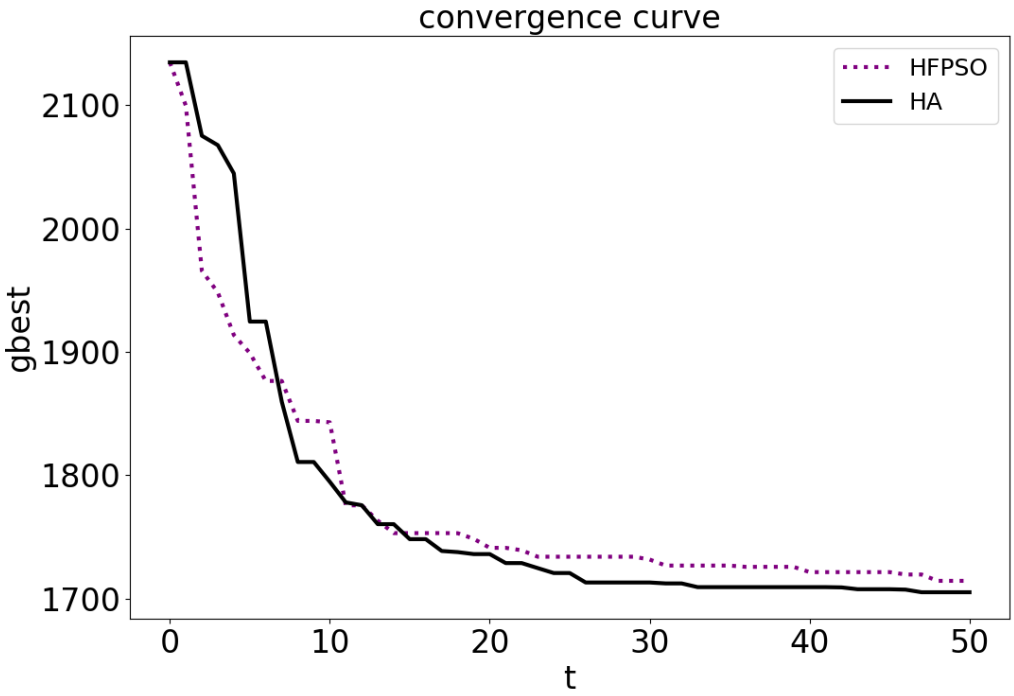


global best evolution curve

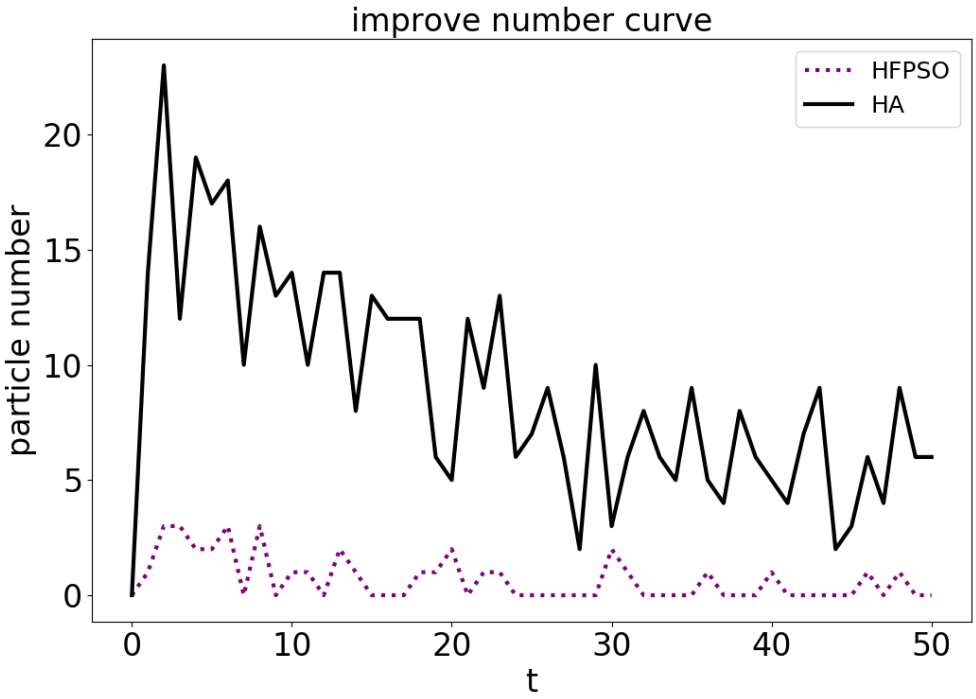


particles' number curve

Figure 4.1: One time run for F_{13}

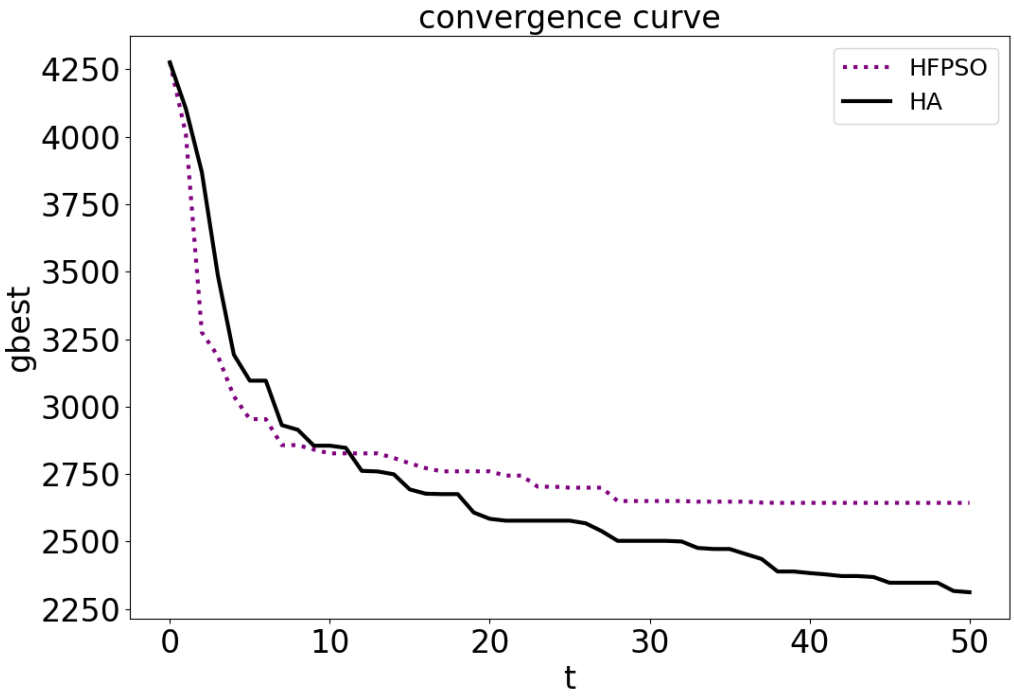


global best evolution curve

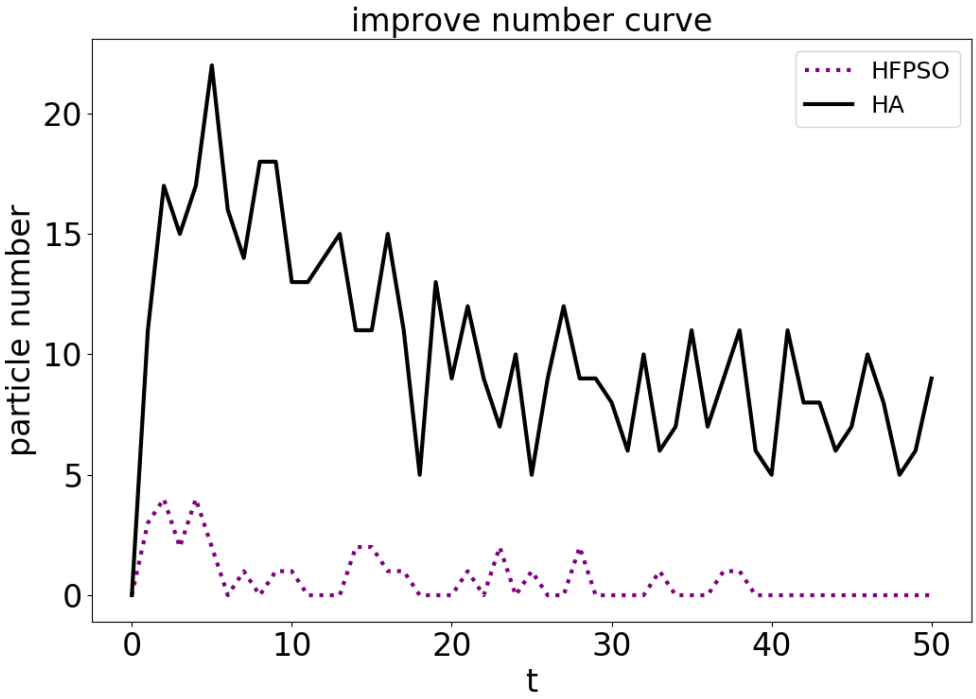


particles' number curve

Figure 4.2: One time run for F_{14}



global best evolution curve

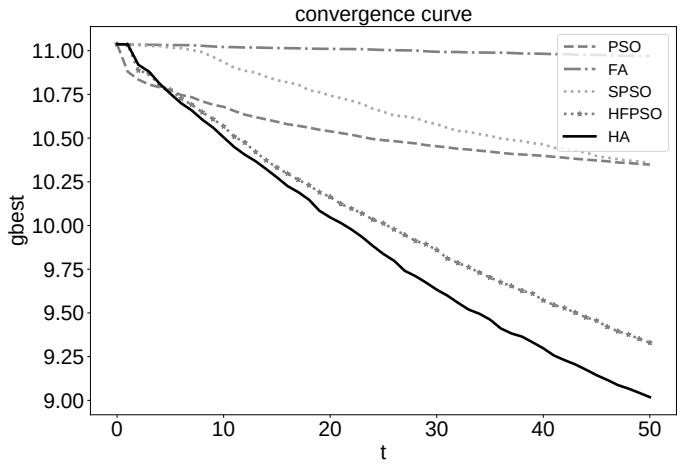


particles' number curve

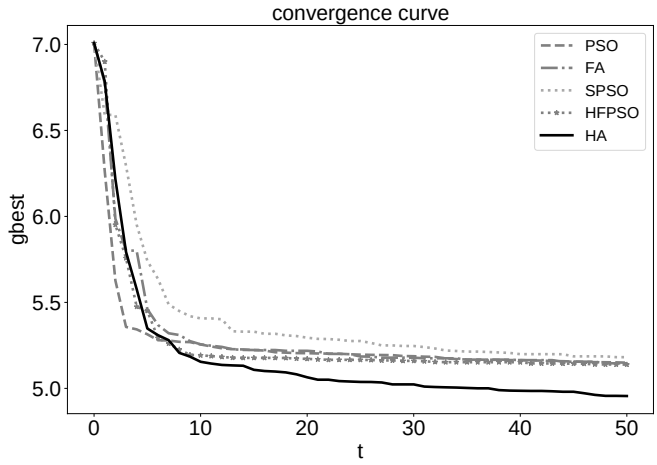
Figure 4.3: One time run for F_{15}

Table 4.2: CEC 17 expensive optimization test problems

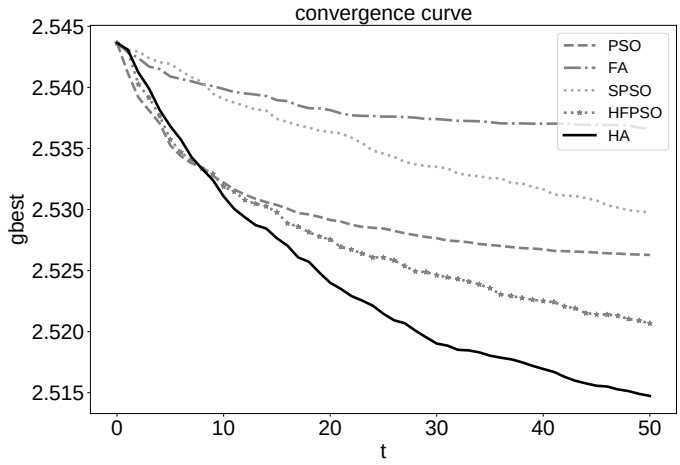
Type	No.	Description	F_i^*
Unimodal functions	1	Shifted and Rotated Bent Cigar Function	100
	2	Shifted and Rotated Sum of Different Power Function	200
	3	Shifted and Rotated Zakharov Function	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock Function	400
	5	Shifted and Rotated Rastrigin Function	500
	6	Shifted and Rotated Expanded Scaffer F6 Function	600
	7	Shifted and Rotated Lunacek Bi Rastrigin Function	700
	8	Shifted and Rotated Non-Continuous Rastrigin Function	800
	9	Shifted and Rotated Levy Function	900
	10	Shifted and Rotated Schwefel Function	1000
Hybrid Functions	11	Hybrid Function 1 (N = 3)	1100
	12	Hybrid Function 1 (N = 3)	1200
	13	Hybrid Function 1 (N = 3)	1300
	14	Hybrid Function 1 (N = 4)	1400
	15	Hybrid Function 1 (N = 4)	1500
	16	Hybrid Function 1 (N = 4)	1600
	17	Hybrid Function 1 (N = 5)	1700
	18	Hybrid Function 1 (N = 5)	1800
	19	Hybrid Function 1 (N = 5)	1900
	20	Hybrid Function 1 (N = 6)	2000
Composition Functions	21	Composition Function 1 (N = 3)	2100
	22	Composition Function 1 (N = 3)	2200
	23	Composition Function 1 (N = 4)	2300
	24	Composition Function 1 (N = 4)	2400
	25	Composition Function 1 (N = 5)	2500
	26	Composition Function 1 (N = 5)	2600
	27	Composition Function 1 (N = 6)	2700
	28	Composition Function 1 (N = 6)	2800
	29	Composition Function 1 (N = 3)	2900
	30	Composition Function 1 (N = 3)	3000



F_1

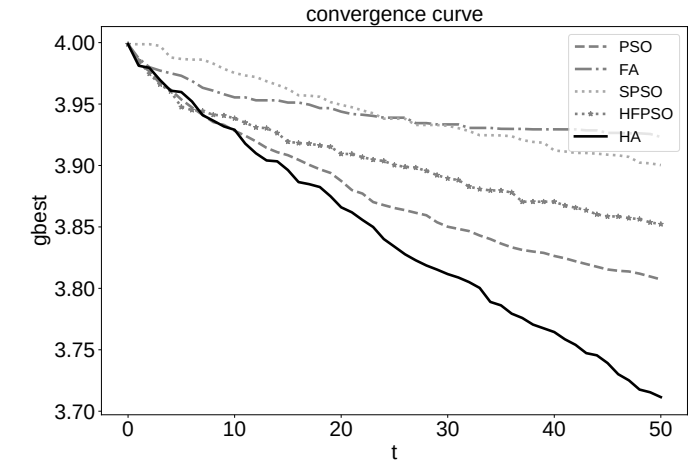


F_2

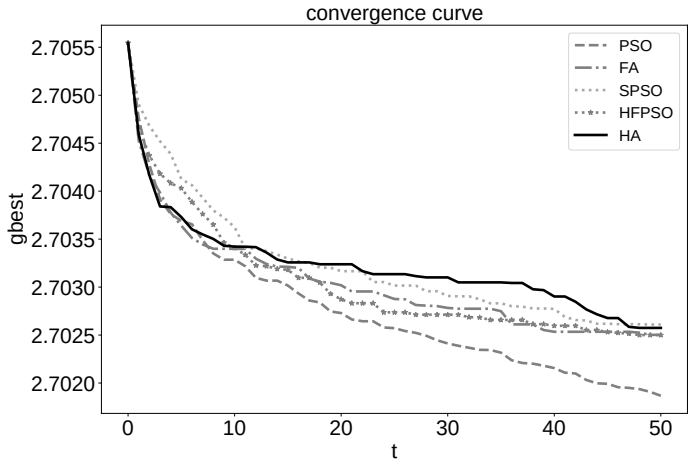


F_3

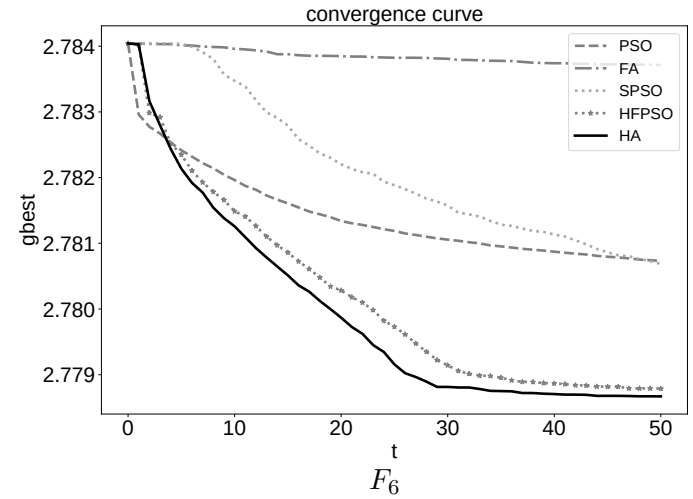
Figure 4.4: CEC 2015 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



F_4

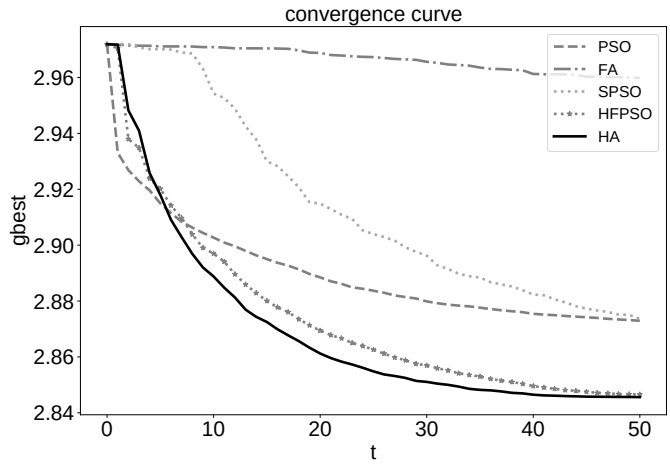


F_5

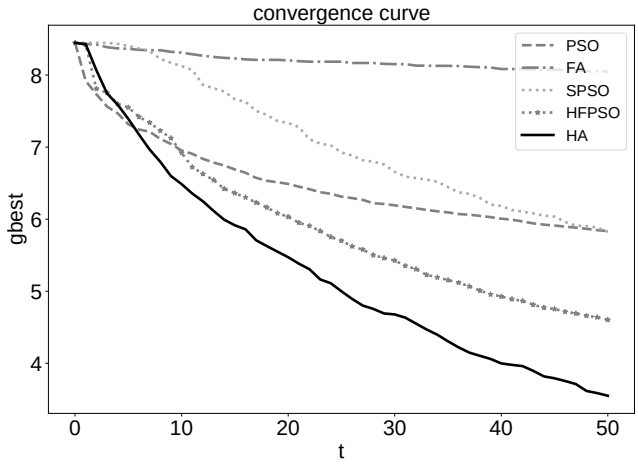


F_6

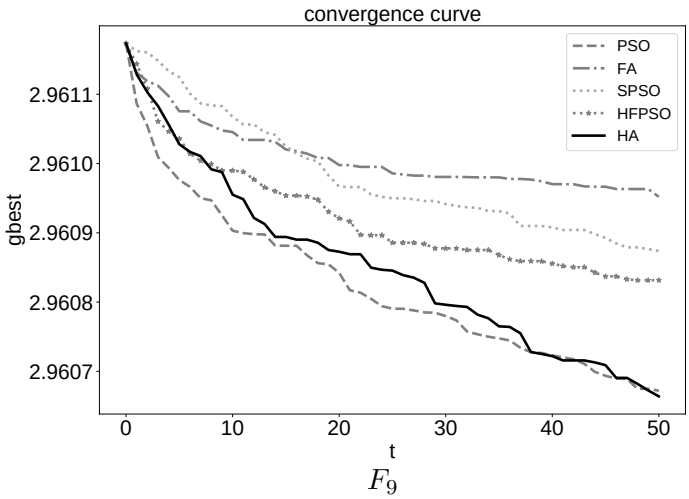
Figure 4.4: CEC 2015 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



F_7

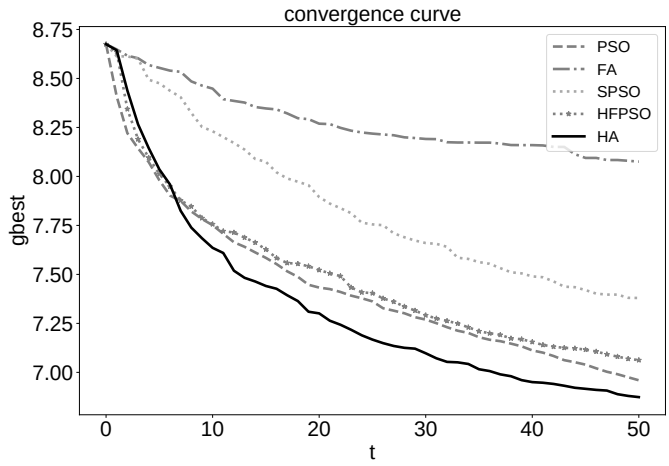


F_8

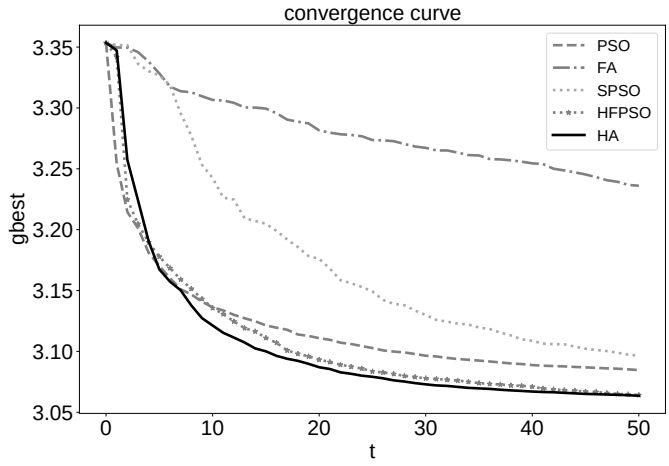


F_9

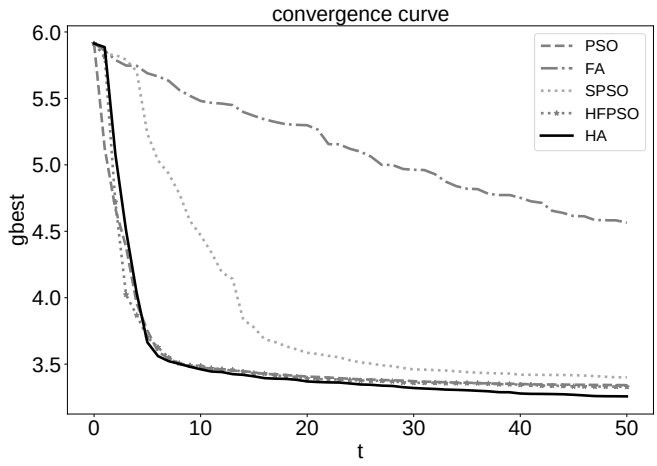
Figure 4.4: CEC 2015 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



F_{10}

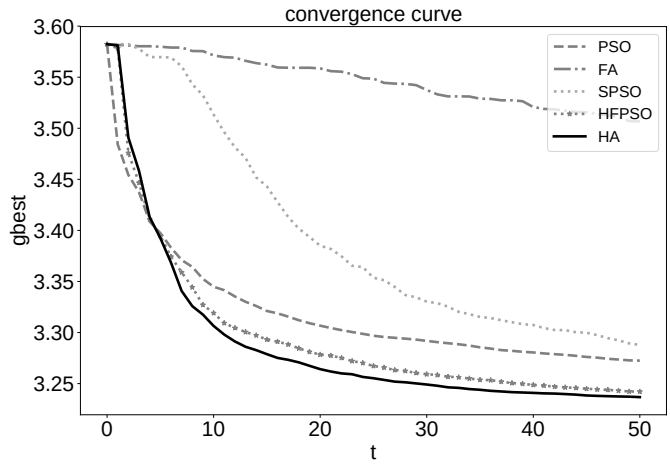


F_{11}

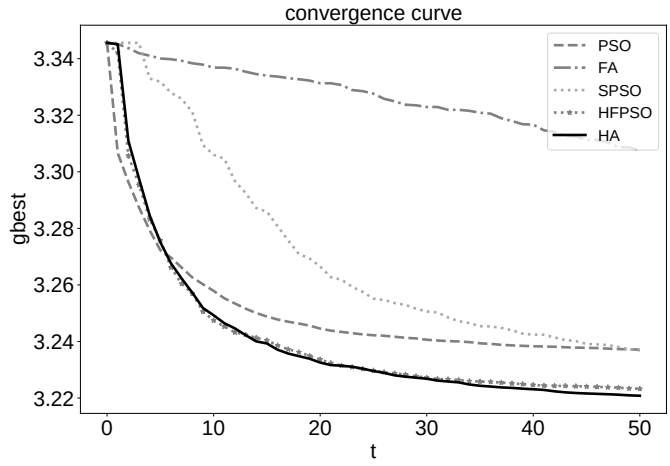


F_{12}

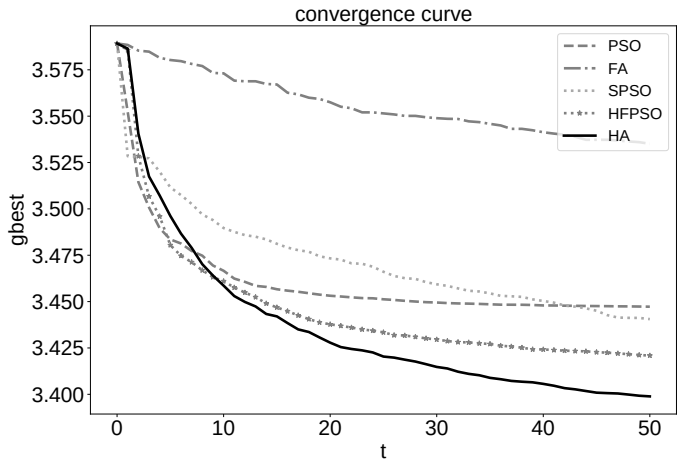
Figure 4.4: CEC 2015 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



F_{13}

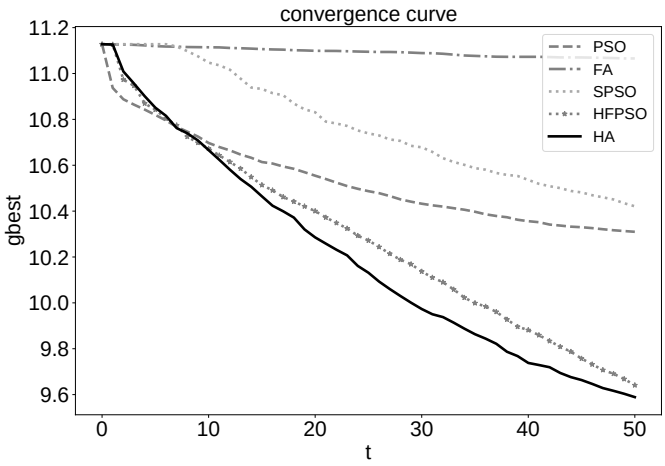


F_{14}

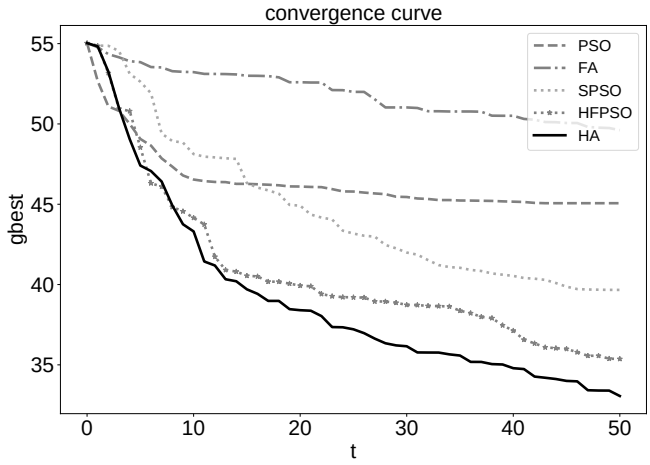


F_{15}

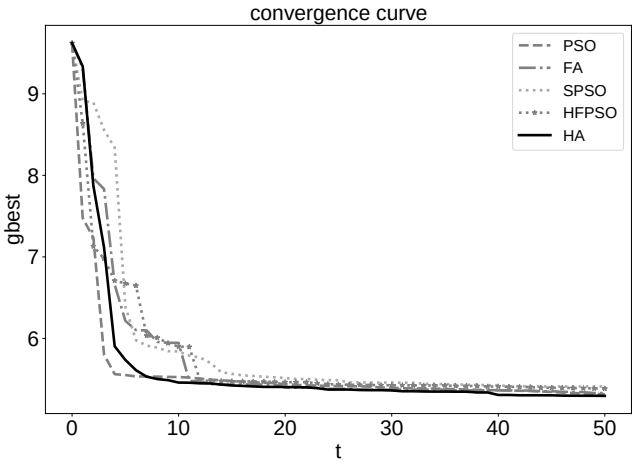
Figure 4.4: CEC 2015 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 1

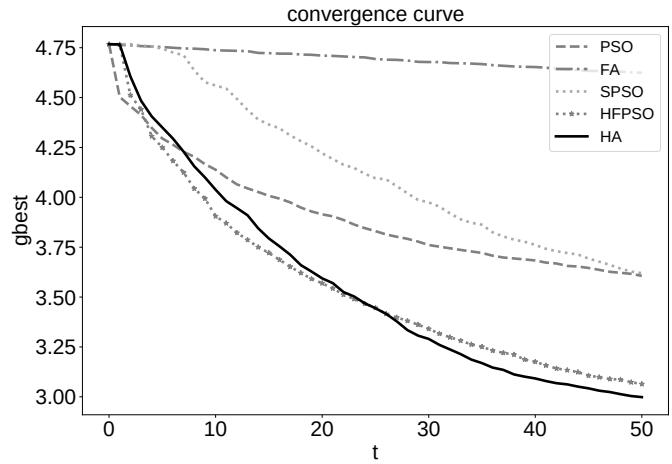


No. 2

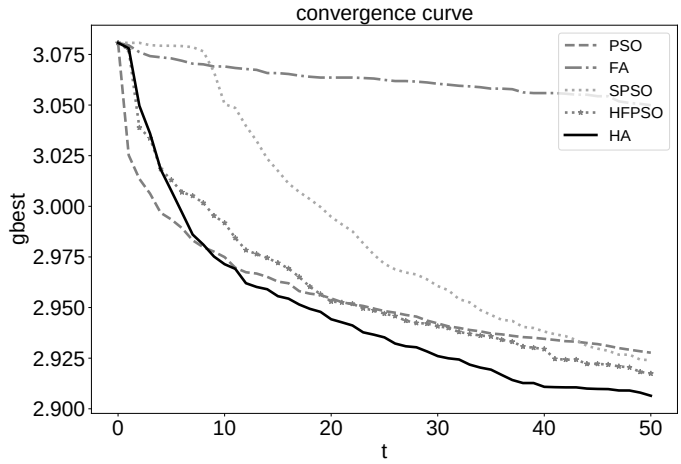


No. 3

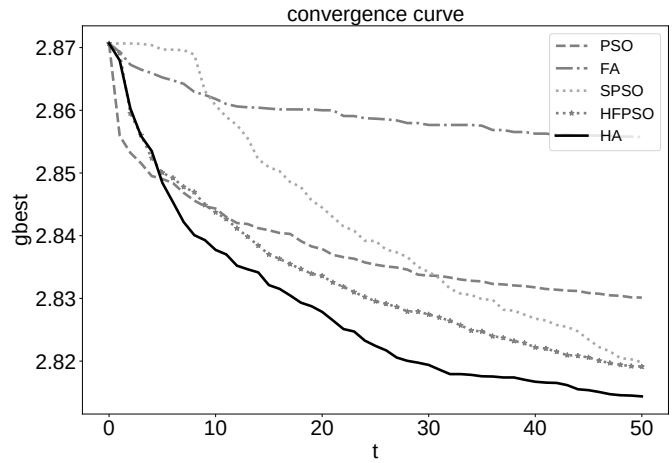
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 4

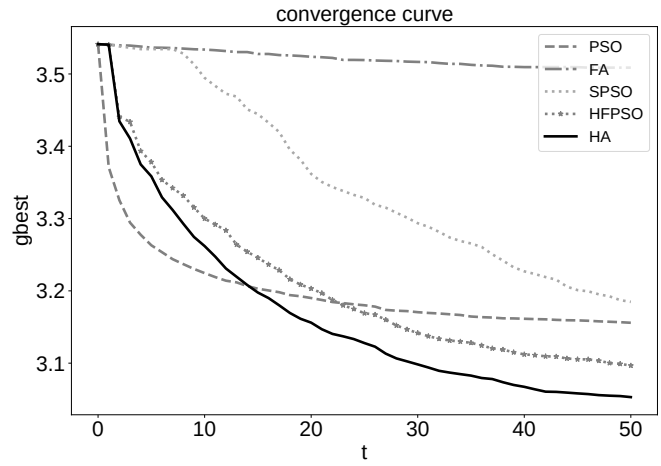


No. 5

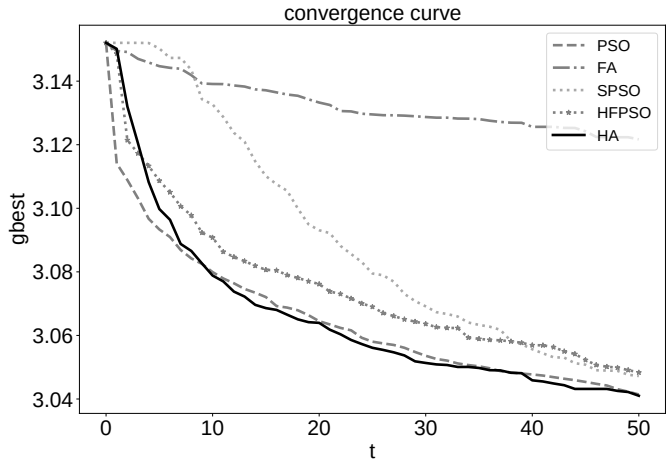


No. 6

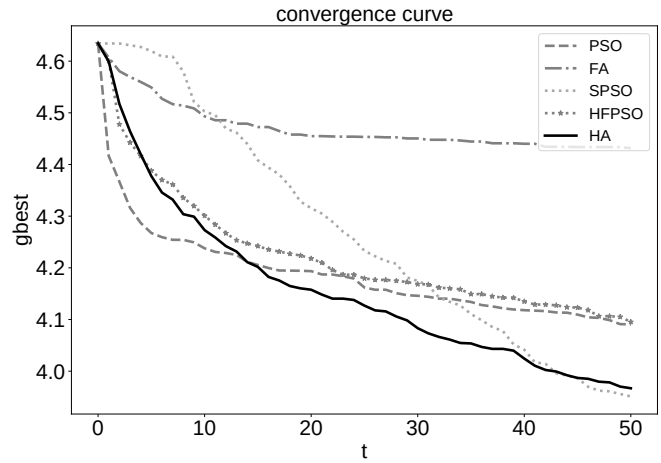
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 7

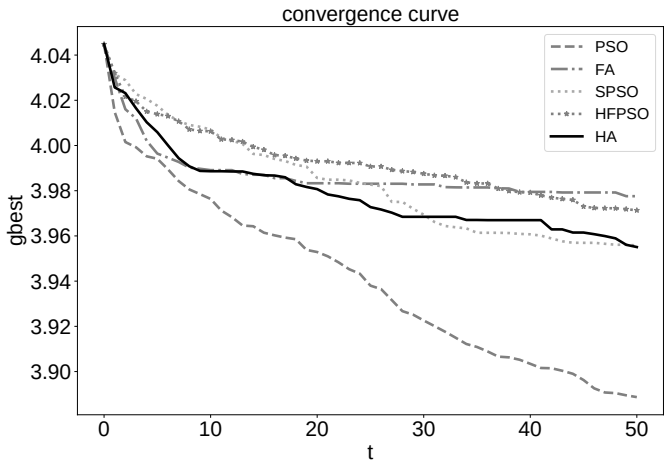


No. 8

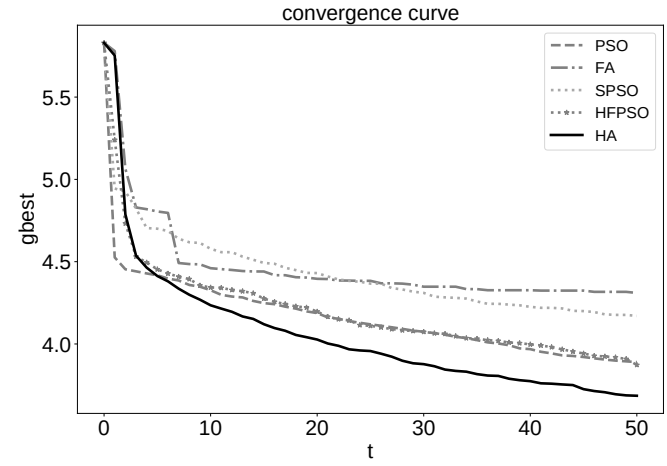


No. 9

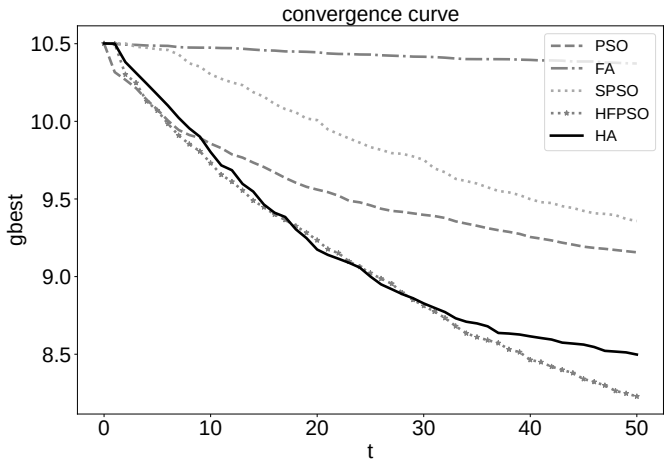
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 10

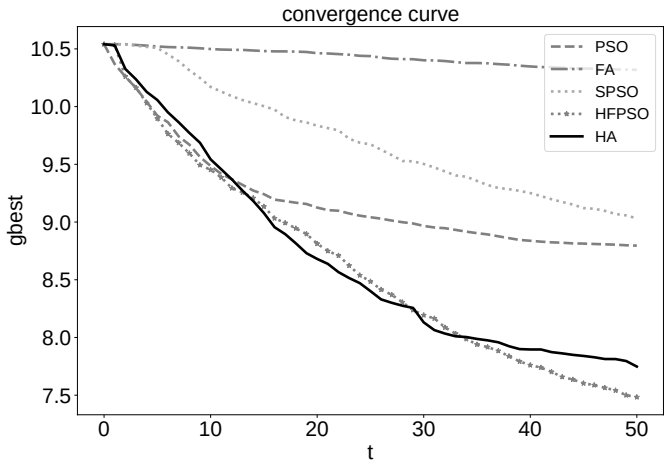


No. 11

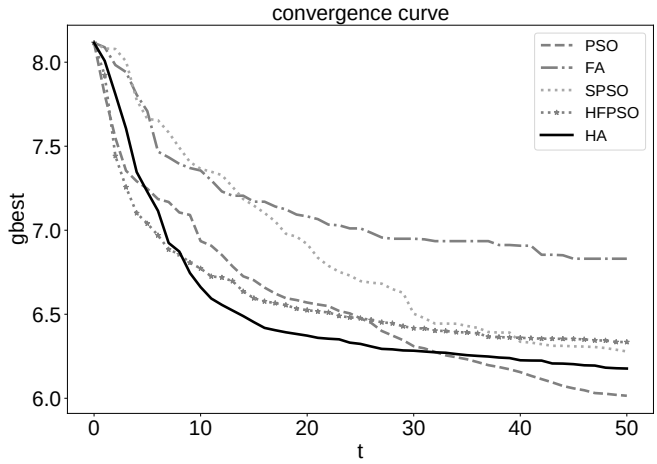


No. 12

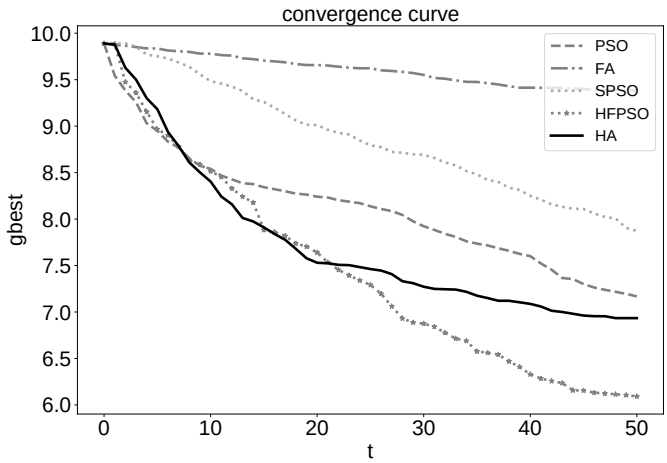
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 13

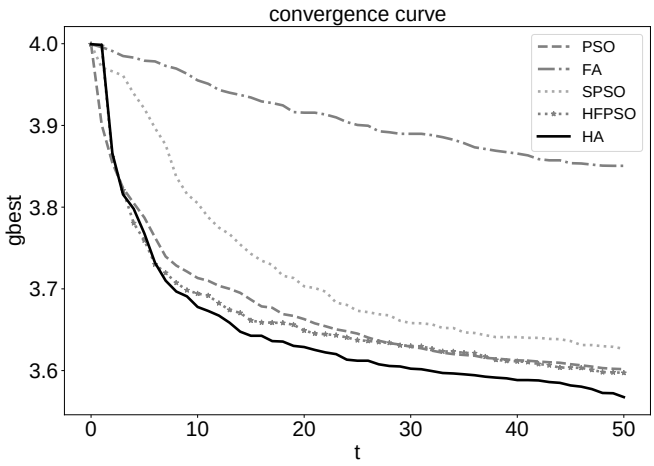


No. 14

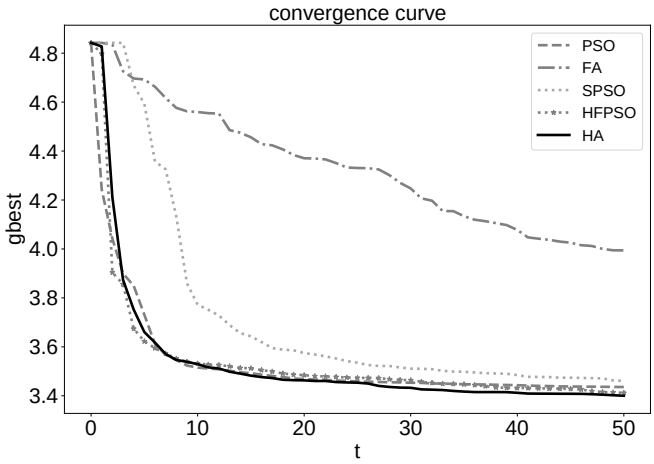


No. 15

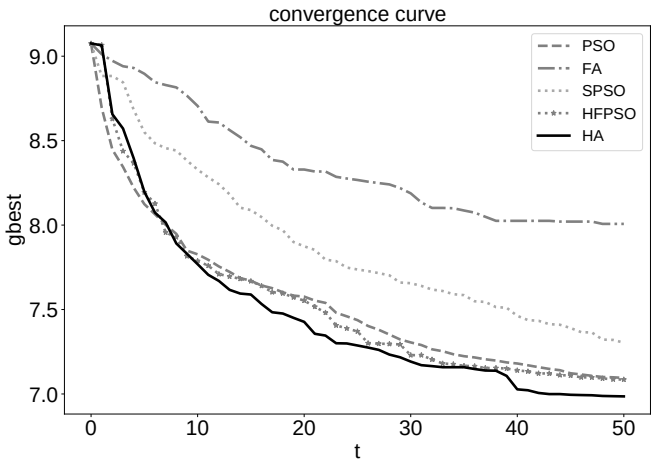
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 16

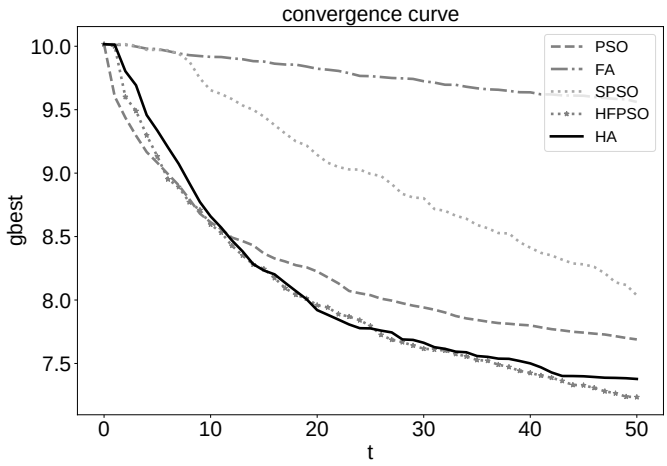


No. 17

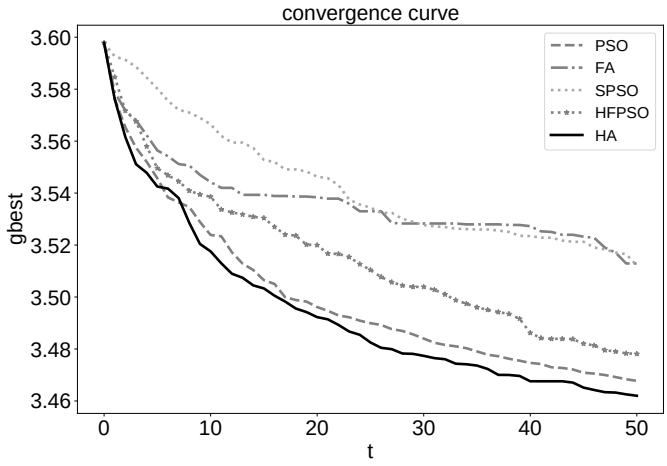


No. 18

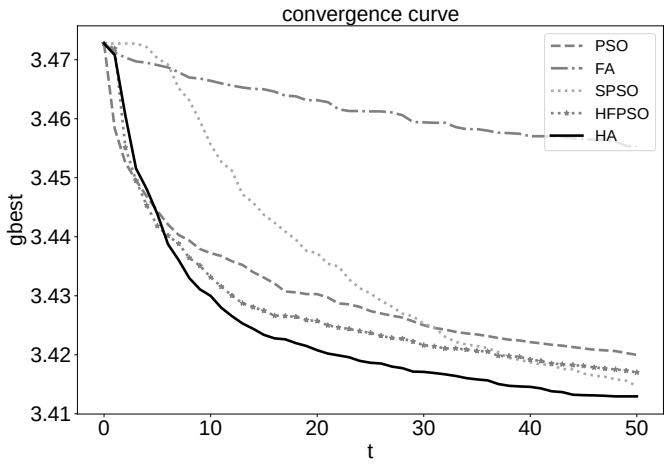
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 19

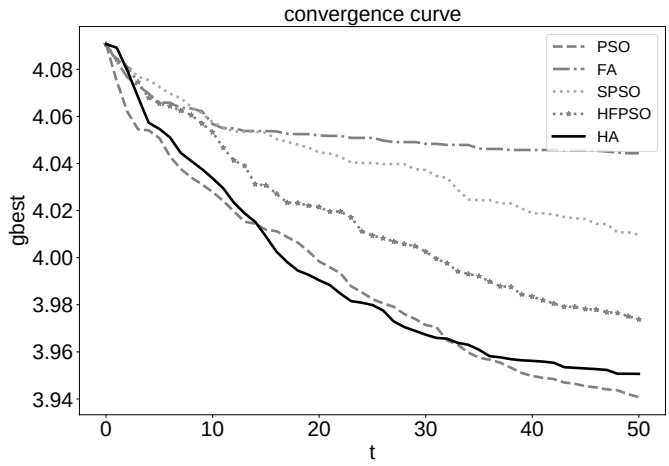


No. 20

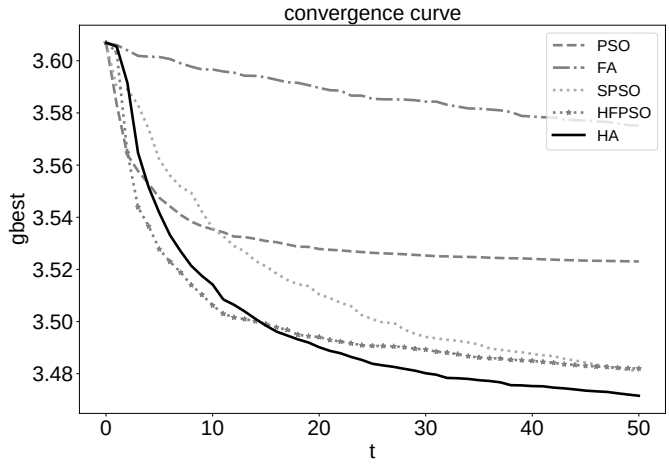


No. 21

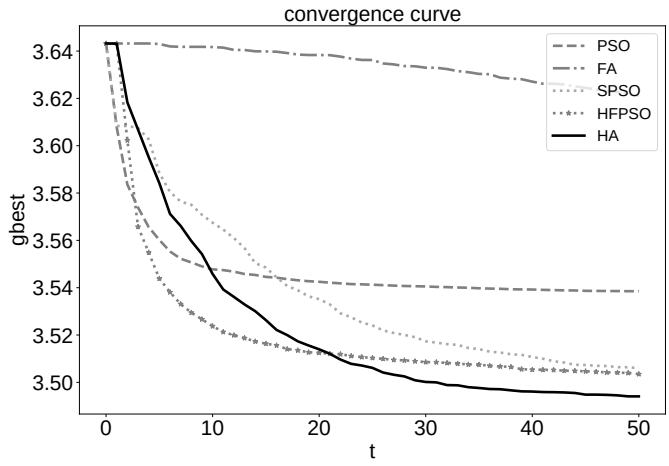
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 22

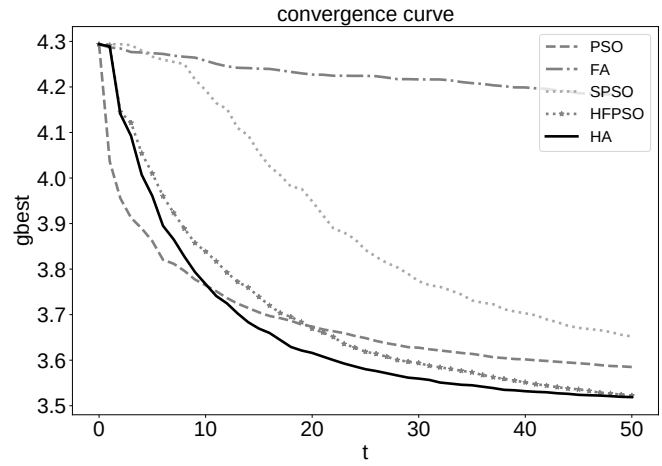


No. 23

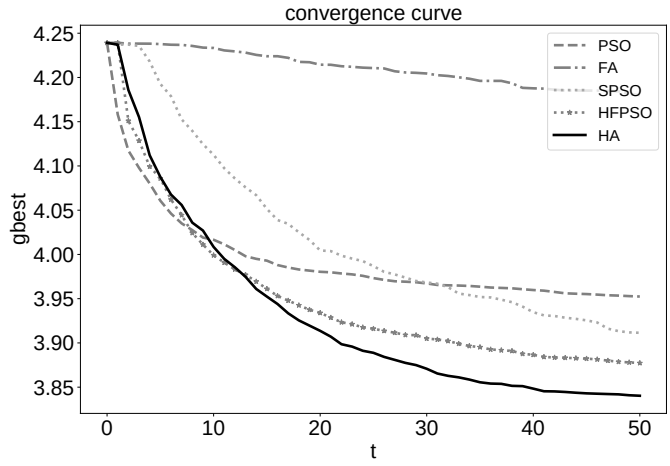


No. 24

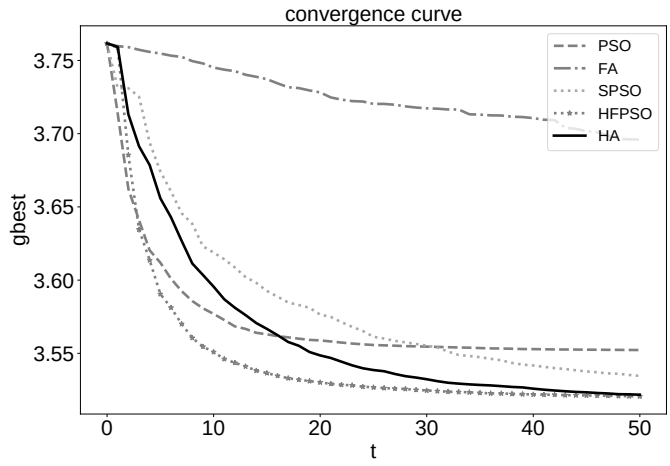
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 25

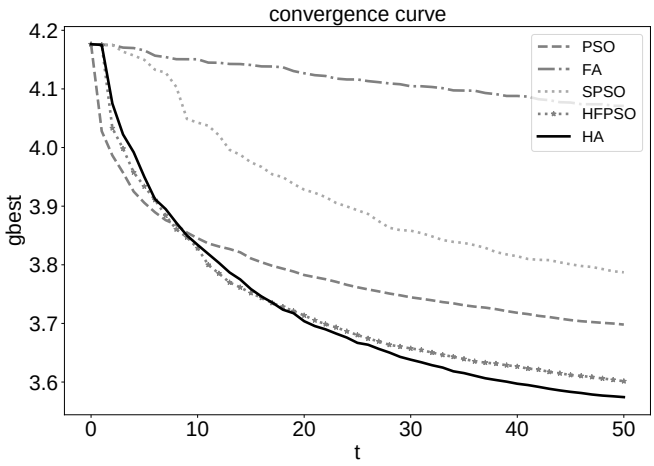


No. 26

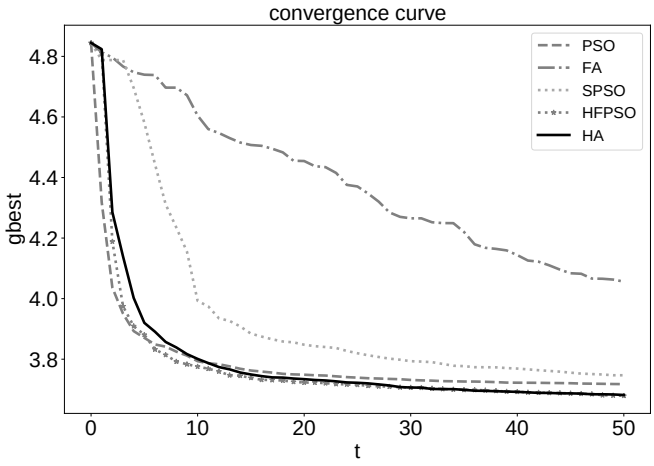


No. 27

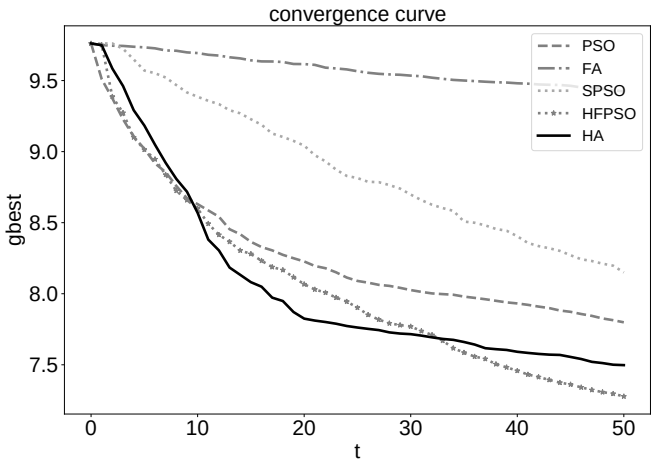
Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)



No. 28



No. 29



No. 30

Figure 4.5: CEC 2017 convergence curve (horizontal axis indicates iteration number, vertical axis indicates global best value of 20 implements' mean result)

landscape. To the F_2 function, it is unimodal with a sensitive direction. For unimodal functions, the model selection helps the hybrid swarm (HA) performs better than PSO, FA, SPSO and HFPSO. It shows that the model selection helps to improve the progress in this problem than others. Then in Figure 4.4, for F_1, F_2, it is able to see that PSO shows a high drop down rate at first and gradually slows down to get close to premature. FA keeps to find better solutions but at a slow improve rate. HA shows a high improving rate in the F_1 function and keep improving trend in F_2 function. With the model selection strategy, the HA keeps high improved rate as PSO even though it loses to PSO at the begin time. The hybrid swarm keeps progress while PSO slow down the improve rate. It means that the hybrid swarm shows good performance in exploration stage or exploitation stage.

F_3 to F_9 are simple multimodal functions. HA gets good rank scores in most of them except the F_5 function. For that multiple local optimums exist, the ability to get out of the traps is important for processing multimodal problems. In Figure 4.4, about F_3, F_6, F_7, F_8, PSO fails for premature while HA is successful to keep high improved rate in them and get better solutions. It shows that the high convergence rate and strong local search both work in the hybrid swarm. The model selection strategy shows good performance in multimodal problems.

For F_{10} to F_{15}, these problems are hybrid functions or composition functions. Since these problems are complicated for they are composited of several basic functions, the model selection mechanism is expected to well perform for such kinds of problems even though the landscapes of these problems are unknown. HA performs best in these functions. It is able to see that HA keeps gradually getting better optimums. In the graph of the F_{15} function, HA still keeps improving the best value after PSO stops improvement. Since the proposed hybrid swarm is composed of PSO and FA, it shows that the hybrid mechanism contributes to improving the premature of PSO.

In the CEC 2017 benchmark, there provide 30 functions. The result of the

benchmark shows in Figure 4.5, Table 4.4 and Table 4.6. HA performs best in the unimodal problems. For the multimodal problem No.4 - No.10, HA performs best in most of them except No.9, No.10 function. It means that the hybrid swarm with the model selection mechanism shows good performance while processing multimodal problems. To the complex hybrid functions and composition functions No.11 - No.30, HA performs best in most of them. From the whole result, HA gets an average score which is far better than others.

Based on the above, as an optimize solver, the hybrid swarm composed of PSO, FA is proposed and uses the model selection strategy for improving the performance for calculating black-box optimization problems. The test result shows that HA gets the best score in the CEC 2015 benchmark and CEC 2017 benchmark. Besides, the hybrid swarm performs both high convergence speed and the ability to get better optima when comparing it to PSO and FA. The model selection strategy is helpful to improve the hybrid swarm's performance compare to others.

4.3 Statistical test

From the test result of CEC 2015 and CEC 2017, the proposed hybrid gets the best average rank score in Table 4.5 and Table 4.6 and that means the proposed hybrid algorithm performs best. To verify that the performance of the proposed algorithm is statistically different from other algorithms, the Holm-Bonferroni procedure [17, 22] is used for analyzing the performance difference within PSO, FA, SPSO, HFPSO and the proposed hybrid algorithm. In this statistical test, the null hypothesis (two algorithms are no different) and the alternative hypothesis (two algorithms are different) are considered when handling the benchmark suites. If the null hypothesis is rejected, there is a significant difference between the performances of two algorithms.

Based on the null hypothesis (two algorithms are no different), the control

method (best performed algorithm) would compare with other $N_A - 1$ algorithms for N_{TP} target problems. Then there is a family of hypotheses that related to the focused method. If the null hypothesis is rejected, there is apparent difference in this pair. For each pair of the hypothesis, there are two samples. x_1, x_2. Let $\overline{x_1}$, $\overline{x_2}$ describe the average rank score of these two samples. The variance of the two samples are s_1, s_2, and the sample number of them are n_1, n_2. For analyzing the differences of these two samples, the test statistics z is given as:

$$z_j = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{s_1}{n_1} + \frac{s_2}{n_2}}} \tag{4.2}$$

In this test, the average rank score of two algorithms are two samples. Indicate the average rank score of algorithm j and 0 as $\overline{x_1}$, $\overline{x_2}$ and use S_j, S_0 to describe the variance of them. The average rank scores are used for calculating the difference of the samples' mean value. Depends on the Friedman test, the variance of two algorithms: s_1, s_2=$N_A(N_A + 1)/12$, n_1, n_2=N_{TP}. Put these into formula Eq 4.2, then it is able to get the Formula Eq 4.3.

$$z_j = \frac{S_j - S_0}{\sqrt{\frac{N_A(N_A+1)}{6N_{TP}}}} \tag{4.3}$$

For the ranked algorithm j, the z_j [13] is used to calculate the cumulative normal distribution p value of it. In this test, the level of confidence δ is used as 0.05. then a threshold is calculated as $\theta = \delta/j, j > 0$. Here, "$p < \theta$" indicates the null hypothesis is rejected or it will be accepted.

The statistical test result is shown in Table 4.7 and Table 4.8. HA gets the best rank score in CEC 2015 benchmark, CEC 2017 benchmark. The best rank score owner is selected as a base one for comparison then the rank score of HA is utilized as S_0. Calculating the z value of PSO, FA, SPSO, and HFPSO based on Eq 4.3. Then verify whether "$p < \theta$" to decide to reject or accept that there is no difference between them and proposed HA.

Table 4.3: CEC15 30D result (bold values are the best ones)

	F_1	F_2	F_3	F_4	F_5
PSO	2.2295E+10	1.4104E+05	3.3596E+02	6.4161E+03	**5.0335E+02**
FA	9.3342E+10	1.3989E+05	3.4403E+02	8.3790E+03	5.0408E+02
SPSO	2.2763E+10	1.5188E+05	3.3866E+02	7.9491E+03	5.0421E+02
HFPSO	2.1371E+09	1.3755E+05	3.3165E+02	7.1158E+03	5.0408E+02
HA	**1.0439E+09**	**9.0280E+04**	**3.2714E+02**	**5.1460E+03**	5.0417E+02
	F_6	F_7	F_8	F_9	F_{10}
PSO	6.0358E+02	7.4632E+02	6.8400E+05	9.1342E+02	9.1116E+06
FA	6.0773E+02	9.1165E+02	1.1191E+08	9.1401E+02	1.1896E+08
SPSO	6.0351E+02	7.4747E+02	6.6867E+05	9.1385E+02	2.3966E+07
HFPSO	6.0088E+02	7.0234E+02	4.0099E+04	9.1376E+02	1.1549E+07
HA	**6.0072E+02**	**7.0081E+02**	**3.5467E+03**	**9.1341E+02**	**7.4758E+06**
	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
PSO	1.2152E+03	2.1964E+03	1.8728E+03	1.7262E+03	2.8006E+03
FA	1.7221E+03	3.6644E+04	3.2119E+03	2.0301E+03	3.4285E+03
SPSO	1.2487E+03	2.5173E+03	1.9404E+03	1.7252E+03	2.7577E+03
HFPSO	1.1592E+03	2.1357E+03	1.7465E+03	1.6721E+03	2.6358E+03
HA	**1.1576E+03**	**1.8085E+03**	**1.7249E+03**	**1.6627E+03**	**2.5054E+03**

From the statistical result, the proposed HA outperforms than PSO, FA, SPSO, HFPSO in CEC 2015 benchmark, CEC 2017 benchmark. There is a significant difference between HA and other algorithms. That means the proposed strategy for the hybrid swarm apparently improves the performance and averagely well perform in processing these black-box benchmark optimization problems.

Making hybridization is a promising method to improve the performance by suitably combining different models. In this chapter, a PSO-FA hybridization with a model selection strategy is proposed. In the model selection strategy, an event-driven trigger based on whether the personal best information is updated is utilized. This hybrid swarm aims to improve the weakness as premature by importing different swarm intelligence models' characteristics into a common swarm. This work contributes to building a hybridization of basic PSO and FA. By importing an event-driven trigger to automatically control the personal study process of each individual, the proposed method is easy to be implemented by saving the

cost that controlling the selection process and utilizing the simple PSO and FA as component models. To show how the proposed hybrid swarm performs when comparing it with PSO, FA, SPSO, HFPSO, CEC 2015 benchmark problem and CEC 2017 benchmark problem are used for a comparison test. From the test result, it is able to observe that the proposed hybrid swarm gets the best result in most of the problems. For verifying whether there is an apparent difference between the hybrid swarm with other algorithms, the Holm-Bonferroni procedure is used for a statistical test. From the statistical result, the hybrid swarm with the model selection strategy outperforms others. The PSO-FA hybridization with the event-driven model selection strategy clearly outperforms PSO, FA, SPSO, HFPSO. This chapter introduced an event-driven model selection strategy based on the personal best update to build a PSO-FA hybrid swarm. By changing the status of the individual, the proposed method focuses on the personal study process and contributes to well handle complex problems. This work provides the idea of designing the personal study process in the hybridization.

Table 4.4: CEC17 30D result (bold values are the best ones)

	1	2	3	4	5
PSO	2.0410E+10	1.1607E+45	2.0640E+05	4.0400E+03	8.4665E+02
FA	1.1633E+11	4.0912E+49	2.1394E+05	4.2072E+04	1.1218E+03
SPSO	2.6368E+10	4.6386E+39	2.5781E+05	4.1677E+03	8.3950E+02
HFPSO	4.3752E+09	2.3324E+35	2.4309E+05	1.1596E+03	8.2683E+02
HA	**3.8816E+09**	**1.1477E+33**	**1.9739E+05**	**9.9653E+02**	**8.0619E+02**
	6	7	8	9	10
PSO	6.7629E+02	1.4317E+03	1.1001E+03	1.2318E+04	**7.7391E+03**
FA	7.1732E+02	3.2267E+03	1.3234E+03	2.7010E+04	9.4968E+03
SPSO	6.6041E+02	1.5303E+03	1.1149E+03	**8.9372E+03**	9.0320E+03
HFPSO	6.5935E+02	1.2493E+03	1.1178E+03	1.2454E+04	9.3613E+03
HA	**6.5219E+02**	**1.1299E+03**	**1.0991E+03**	9.2666E+03	9.0174E+03
	11	12	13	14	15
PSO	7.7605E+03	1.4334E+09	6.2467E+08	**1.0351E+06**	1.4717E+07
FA	2.0412E+04	2.3583E+10	2.0836E+10	6.7683E+06	2.4391E+09
SPSO	1.4800E+04	2.2801E+09	1.0757E+09	1.8997E+06	7.4552E+07
HFPSO	7.4597E+03	**1.6923E+08**	**3.0429E+07**	2.1609E+06	**1.2356E+06**
HA	**4.8455E+03**	3.1471E+08	5.5979E+07	1.5028E+06	8.5941E+06
	16	17	18	19	20
PSO	3.9967E+03	2.7269E+03	1.2442E+07	4.8911E+07	2.9360E+03
FA	7.0888E+03	9.8644E+03	1.0163E+08	3.6388E+09	3.2577E+03
SPSO	4.2322E+03	2.8921E+03	2.0126E+07	1.0885E+08	3.2515E+03
HFPSO	3.9560E+03	2.5836E+03	1.2173E+07	**1.7169E+07**	3.0069E+03
HA	**3.6942E+03**	**2.5125E+03**	**9.6756E+06**	2.3869E+07	**2.8973E+03**
	21	22	23	24	25
PSO	2.6301E+03	**8.7253E+03**	3.3347E+03	3.4555E+03	3.8462E+03
FA	2.8531E+03	1.1075E+04	3.7601E+03	4.1805E+03	1.5125E+04
SPSO	2.5983E+03	1.0228E+04	3.0272E+03	3.2061E+03	4.4889E+03
HFPSO	2.6123E+03	9.4143E+03	3.0340E+03	3.1878E+03	3.3297E+03
HA	**2.5879E+03**	8.9259E+03	**2.9618E+03**	**3.1194E+03**	**3.3029E+03**
	26	27	28	29	30
PSO	8.9625E+03	3.5667E+03	4.9891E+03	5.2202E+03	6.2924E+07
FA	1.5138E+04	4.9661E+03	1.1776E+04	1.1382E+04	2.6572E+09
SPSO	8.1553E+03	3.4244E+03	6.1261E+03	5.5838E+03	1.4121E+08
HFPSO	7.5380E+03	**3.3159E+03**	3.9945E+03	**4.7784E+03**	**1.8952E+07**
HA	**6.9230E+03**	3.3246E+03	**3.7522E+03**	4.8066E+03	3.1446E+07

Table 4.5: CEC15 Ranking score (bigger one is better)

	PSO	FA	SPSO	HFPSO	HA
F_1	3	1	2	4	5
F_2	2	3	1	4	5
F_3	3	1	2	4	5
F_4	4	1	2	3	5
F_5	5	3	1	4	2
F_6	2	1	3	4	5
F_7	3	1	2	4	5
F_8	2	1	3	4	5
F_9	4	1	2	3	5
F_{10}	4	1	2	3	5
F_{11}	3	1	2	4	5
F_{12}	3	1	2	4	5
F_{13}	3	1	2	4	5
F_{14}	2	1	3	4	5
F_{15}	2	1	3	4	5
Avg.	3.00	1.27	2.13	3.80	4.80

Table 4.6: CEC17 Ranking score (bigger one is better)

No.	PSO	FA	SPSO	HFPSO	HA
1	3	1	2	4	5
2	2	1	3	4	5
3	4	3	1	2	5
4	3	1	2	4	5
5	2	1	3	4	5
6	2	1	3	4	5
7	3	1	2	4	5
8	4	1	3	2	5
9	3	1	5	2	4
10	5	1	3	2	4
11	3	1	2	4	5
12	3	1	2	5	4
13	3	1	2	5	4
14	5	1	3	2	4
15	3	1	2	5	4
16	3	1	2	4	5
17	3	1	2	4	5
18	3	1	2	4	5
19	3	1	2	5	4
20	4	1	2	3	5
21	2	1	4	3	5
22	5	1	2	3	4
23	2	1	4	3	5
24	2	1	3	4	5
25	3	1	2	4	5
26	2	1	3	4	5
27	2	1	3	5	4
28	3	1	2	4	5
29	3	1	2	5	4
30	3	1	2	5	4
Avg.	3.03	1.07	2.50	3.77	4.63

Table 4.7: Holm-Bonferroni procedure for CEC 2015 30D problems (comparison base on HA1)

	score	z	p	θ	h^*
FA	1.2667E+00	-6.1315E+00	4.3538E-10	1.2500E-02	1(rejected)
SPSO	2.1333E+00	-4.6246E+00	1.8768E-06	1.6667E-02	1(rejected)
PSO	3.0000E+00	-3.1177E+00	9.1137E-04	2.5000E-02	1(rejected)
HFPSO	3.8000E+00	-1.7321E+00	4.1632E-02	5.0000E-02	1(rejected)
HA	4.8000E+00				
*: 0-accepted, 1-rejected					

Table 4.8: Holm-Bonferroni procedure for CEC 2017 30D problems (comparison base on HA1)

	score	z	p	θ	h^*
FA	1.0667E+00	-8.7447E+00	1.1183E-18	1.2500E-02	1(rejected)
SPSO	2.5000E+00	-5.2174E+00	9.0720E-08	1.6667E-02	1(rejected)
PSO	3.0333E+00	-3.9192E+00	4.4425E-05	2.5000E-02	1(rejected)
HFPSO	3.7667E+00	-2.1311E+00	1.6542E-02	5.0000E-02	1(rejected)
HA	4.6333E+00				
*: 0-accepted, 1-rejected					

Chapter 5

Multifactorial hybrid swarm

Conventional algorithms focus on providing an actual solution for a single optimization task. These algorithms can also be used to handle the multi-objective optimization problems where the Pareto optimal set is approximated by the population or the archived candidates. In recent years, a novel field of evolutionary computation called evolutionary multitask optimization or evolutionary multitasking has been proposed and utilizing a parallel processing ability based on a population-based search to handle multiple problems separately [21]. It has received attention in the evolutionary computation community. Furthermore, for processing multiple optimization problems with a single population, multifactorial optimization has been presented as a method in which each constitutive task would affect the evolution of the population [21]. As an actual multifactorial optimization framework that is inspired by the bio-cultural models of multifactorial inheritance, a multifactorial evolutionary algorithm (MFEA) is developed [12]. In MFEA, a skill factor is used to assign a preferred task for each individual of the population. This method is proposed to enhance productivity for effectively dealing with industrial problems. It is expected to utilize the latent correlations among instinct tasks by using multifactorial optimization.

On the other hand, swarm intelligence is one of the popular population-based metaheuristics for optimization problems. Besides, making hybridization of suit-

able selected swarm intelligence models is also one way for processing optimization problems effectively. Previous works show that hybridization can improve the performance of processing optimization problems.

This chapter introduces a multifactorial PSO-FA hybrid algorithm (MFHA) [56]. The PSO-FA hybrid swarm utilizes the model selection mechanism with an event-driven trigger based on whether the personal best information is updated. In the proposed multifactorial hybrid swarm, a skill factor is used to help the task assignment of individuals that would help individuals in the population to explore different tasks. For a multifactorial swarm, it is important to suitably assign a task to each individual in the population. This assignment is usually based on the skill factor that indicates the preferred task to each individual based on the pre-evaluated ranking and is exchanged by a crossover in the evolutionary algorithm. Since there is no explicit exchange mechanism in the swarm intelligence model, thus a skill factor reassignment is introduced in this chapter. The hybrid swarm is expected to improve the best solutions in multiple tasks with a combination of a model selection mechanism and skill factor reassignment. By carrying out a numerical experiment of the multifactorial PSO (MFPSO) and the multifactorial PSO-FA hybrid algorithm on the benchmark multifactorial optimization problem, then comparing them with each other to show how the hybrid swarm improves performance than PSO from a total evaluation.

5.1 Multifactorial optimization

Evolutionary multifactorial optimization is a methodology to realize evolutionary multitasking, based on the population-based search. It simultaneously deals with multiple optimization tasks. Here, K represents the number of tasks and $T_k, k = 1, 2, \ldots, K$ describes each target task. Then, X^k denotes the search space and F^K denotes the objective function of the corresponding task, respectively. Assume that all optimization tasks are the minimization task without loss of generality,

and let $x^1, x^2, \ldots, x^K$ be the optimal solutions of the corresponding optimization task, where $x^k = argmin_{x \in X^k} \left\{F^k(x)\right\}$. By iterative evaluating solutions in each task, evolutionary multifactorial optimization aims to get whole tasks' solutions as,

$$x^1, x^2, \ldots, x^K = argmin \left\{F^1, F^2, \ldots, F^K\right\} \tag{5.1}$$

A unified search space is provided for the search population. Here use a set $P = p_1, p_2, \ldots, p_N$ to describe individuals in the population. In the search population, one individual p_i is represented by a D dimension vector. For each individual $p_i, i = 1, 2, \ldots, N$, it would be encoded in a unified space which encompasses $X^1, X^2, \ldots, X^K$, and decoded in each task as $x_i^1, x_i^2, \ldots, x_i^K$, where $x_i^k \in X^k$. Describe the dimensionality of these tasks as $D_k, k = 1, 2, \ldots, K$. Then the dimension of the unified space would be defined as $D = max \{D_k\}$. The task k should be handled in X^k, that means the number of D_k variables would be selected out from the vector of the individual in D dimensional space. As a simple implementation, the first D_k variables of the vector can be used to calculate the solution x_i^k in task k. By evaluating the solutions in each task and ranking the result, it is able to get each task's candidate for the solution as in Eq (5.1).

Inspired by multifactorial bio-cultural models, the multifactorial evolutionary algorithm (MFEA) [12] is developed. In the MFEA, it utilizes a skill factor that is used to assign preferred tasks for individuals and is defined for deciding which task the individual performs best in one generation. It is able to get the skill factor as below: Get multiple evaluation fitness by evaluating x_i^k in each task's definition space. For each task $T_k, k = 1, 2, \ldots, K$, it is able to get each individual's fitness $f_{ik}(\cdot)$. By comparing and ranking the fitness in the task, it is able to get the rank value r_{ik} for each individual following the ascending order of the ranking result. So the task index in the rank value is set as the individual's skill factor ($\tau_i = argmin_k \{r_{ik}\}$). Then one individual would be evaluated only

in the preferred task after assign a preferred task to it. Here individuals own the same skill factor are divided into sub-swarm $P^k, k = 1, 2, \ldots, K$. Where j^k denote the index that has skill factor $\tau_j = k$ and J^k be a set of index j^k, then $P^k = \left\{p_i | i \in J^k\right\}, k = 1, 2, \ldots, K$. Note that $\cup_{k=1}^{K} P^k = P$. There are different methods to decide the skill factor for one individual. In fact, it also utilizes a sub-population model in MFEA, and genetic operators such as crossover and mutation generate migration. In MFEA, it randomly generates the skill factor of the offspring based on parents' skill factor. In this work, ranking evaluation values would be used for deciding the skill factor of individuals. For the multifactorial swarm intelligence model, the skill factor reassignment time should be considered since there is no automatic exchange mechanism in the population.

It is important to consider the relationship among these tasks as the degree of intersection of the global optima and the correspondence in the fitness landscape. Utilizing the inter-task relationship is significant to contribute to the effectiveness of the search. For efficiently utilizing the relationship of distinct optimization tasks, evolutionary multitasking optimization is proposed. Automatic search helps the knowledge transfer among different problems. By applying swarm intelligence to multifactorial optimization, the multifactorial swarm is able to be developed.

In recent years, swarm intelligence model has been applied to multifactorial optimization for developing swarm intelligence based multifactorial optimization method. As an example, by applying brain storm optimization (BSO) algorithm to multifactorial optimization, a multifactorial brain storm optimization algorithm (MFBSA) has been introduced in the previous study [66]. This method proposed applying clustering technique into multitasking. Clustering the population by setting the optima of each task as the cluster center. Then probably generating the offspring of individuals in each task.

There are other swarm intelligence models as the particle swarm optimization applied to the multifactorial optimization. Then in the previous study, a multifactorial optimization based on particle swarm optimization is introduced and aim

to utilize particle swarm optimization to enhance the convergence effect [57]. It shows that the particle swarm operator can effectively accelerate the convergence on some benchmark problems.

There are also some methods to improve the multifactorial PSO then an adaptive multifactorial particle swarm optimization is proposed in the previous study [46]. Different from some other multifactorial PSO which utilizing use a fixed inter-task learning in the evolution process, an inter-task learning-based information transferring mechanism is designed in the adaptive multifactorial particle swarm optimization. The mechanism uses a differential term and an inter-task crossover to help the particles explore a broad search space.

5.2 Multifactorial PSO-FA hybrid algorithm

Multifactorial optimization has the feature of transferring relevant knowledge to simultane-ously accelerate convergence towards near-optimal solutions of multiple optimization tasks. PSO has the weakness in high-dimensional problems for that the search performance deteriorating in local search stage. When applying PSO to the multifactorial optimization to simultaneously process multiple tasks, particles would gather fast for each task. That may cause the PSO to stop progress fast in multiple high dimension optimi-zation tasks. The proposed hybrid is developed to improve the ability to search for better solutions. By applying the proposed hybrid to multifactorial optimization, it may improve the progress of the global best of multiple tasks.

This section introduces a multifactorial PSO-FA hybrid algorithm which is built with PSO [30] and FA [61]. By updating the individual position p_i of the i-th individual in discrete time t, it is able to search in a continuous space.

5.2.1 Multifactorial particle swarm optimization

There are some methods of multifactorial particle swarm optimization have been proposed in previous works [46, 57]. As a simple application of PSO to the multifactorial optimization, MFPSO is built in this paper. In PSO, previous information is used for updating the status. For each one in the swarm, its personal best position would be recorded for the cognition process. The personal best information means one individual could memorize its previous experience and utilize the memory to contribute to the progress of itself. This factor causes one particle's behavior been affected by its memory. Then in the MFPSO, the personal best is recorded in different tasks. For individual i, the personal best in each task k is recorded as pb_i^k. Besides, social common knowledge as the global best position is recorded and shared among whole individuals. The group-shared information contributes to the whole group's communication and is updated by the leader individual. Then the common knowledge also plays a role in building one individual's activity. In MFPSO, the global best in task k is described as gb^k. The personal work and the social work are combined with an inertia factor to construct one individual's moving velocity in PSO. Through some random factors, it is able to control the effect of knowledge learning.

For each individual, the information update occurs in one task. In the initial step, the individual would be evaluated for whole tasks to get the skill factors. Then each individual would be evaluated in only one preferred task in one step. Therefore, in one step, only one task's fitness for the individual is able to be updated. From the above, the personal best information is selected to be used for updating the skill factor. Let pb_i^k be the personal best position of i-th particle in task k and gb^k be the global best position of task k, respectively. That is the global best position is a shared memory of P^k. MFPSO's update equations were defined as Eq (5.2):

$$\begin{cases} v_i(t+1) = wv_i(t) + c_1 r_1(pb_i^k(t) - p_i(t)) + c_2 r_2(gb^k(t) - p_i(t)) \\ p_i(t+1) = p_i(t) + v_i(t+1) \end{cases} \tag{5.2}$$

Here, w is an inertia weight, c_1, c_2 are constants, r_1, r_2 are randomly selected from the uniform distribution over [0,1].

5.2.2 Firefly algorithm for multifactorial optimization

By applying the firefly algorithm to the multifactorial optimization, the firefly algorithm is able to process multiple optimization tasks simultaneously. In the application of the multifactorial optimization, individual $p_i \in P^k$ would compare brightness with another individual $p_j, (p_j \in P^k, j \neq i)$, and a set that elements are brighter fireflies than i in task k is denoted by $P_i^k \subset P^k$. That means one individual would only be attracted by the agents who prefer to the same task. So each individual would compare with other individuals in P^k except itself. It should be considered that the brightness will be changed with the distance r_{ij}. So attractiveness would be varied based on the Euclidean distance between $p_i(t)$ and $p_j(t) \in P_i^k$. Light intensity at r distance is described as $I = I_0/r^2$ (I_0 is the intensity of the light source). For each firefly, light is absorbed and affected by a constant light absorption coefficient.

Then, FA's update equation is proposed as Eq (5.3).

$$p_i(t+1) = p_i(t) + \sum_{p_j \in P_i^k} (\beta \exp[-\gamma r_{ij}^2](p_j(t) - p_i(t)) + \alpha \varepsilon_t) \tag{5.3}$$

Here, β indicates an attraction weight to the brighter fireflies, and γ represents the light absorption rate according to the distance. r_{ij} means the distance between i and j. Then, in the last term, α is a parameter for controlling the step size, and ε_t is a random vector for a random walk whose elements are sampled from Gaussian distribution with 0 mean and unit variance.

5.2.3 Multifactorial hybrid algorithm

This work introduces an application of the model selecting PSO-FA hybrid algorithm to multifactorial optimization. When processing a single problem, the hybrid swarm utilizes a model selection strategy to select either PSO or FA model for one individual. In this strategy, whether the personal best is updated becomes a trigger of model selection. Thus, if the personal best position was updated then the particle selects to move with the local search of the FA model otherwise it moves as the PSO model.

In the hybrid swarm, each particle records its personal best and the whole group shares the group best that is selected from the personal best of the particles. When one individual moves as a firefly, it compares the intensity with other individuals for updating moving velocity. By applying the skill factor into the PSO-FA hybrid algorithm, the multifactorial PSO-FA hybrid algorithm (MFHA) is provided for simultaneously processing multiple optimization problems with a single swarm. In this algorithm, the population would record the evaluation result in each task. That means individuals would record the information by tasks. By comparing the recorded function fitness of the individuals, each one in the population gets its skill factor. One individual would only update its preferred task's fitness record in one iteration. That means it needs a variable to control the update time of the skill factor, which is defined as η [55]. This parameter is used to control the task assignment time of individuals. Besides, when the skill factor reassignment occurs, one individual's skill factor may be changed by computing and comparing the result of recorded fitness. If one individual's preferred task is changed when skill factor reassignment occurs, then it would be moved into another group. The individuals who own the same skill factor value would be divided into the same group for comparison. For the individuals who prefer the same task, they share the global best in the preferred task. At the same time, one individual utilizes its personal best in the preferred task. When one works as a firefly, it would only compare the intensity with the individuals in the preferred

Table 5.1: Pseudo code of PSO-FA hybrid algorithm

Step1	Initialize the individual position of swarm P
Step2	Evaluate each individuals fitness in K tasks and
Step3	Set its personal best pb_i^k,$k = 1, 2, \ldots, K$ for each individual
Step4	If terminate condition not meet
	Find the global best gb^k for each task from pb_i^k, $(i = 1, 2, ..., N)$
	At every η step, assign skill factor for each individual based on pb_i^k
	For each individual i
	Get the skill factor of it as k
	If the personal best memory is not updated last time
	Move it as particle according to pb_i^k and gb^k
	Else
	Move it as firefly in comparing with p_j in P_i^k
	Update fitness in task k, and compare with pb_i^k
	If fitness is improved
	Update pb_i^k and set a model change flag
Step5	End

task. The multifactorial PSO-FA hybrid algorithm is built as in Table 5.1. With the multifactorial optimization structure, the hybrid swarm is expected to well process multiple optimization tasks simultaneously.

In this hybrid algorithm, changing the skill factor value of one particle can reassign its preferred task. That means there needs a variable to update the skill factor. This parameter is used to contribute to the assignment to tasks. Besides, whether the skill factor is updated would be controlled by computing and comparing the result of recorded fitness. In the group for one task, particles and fireflies may exist at the same time. When the individuals perform as the PSO model, they share the common global best memory information and keep all information for each task. While they perform as the FA model, the attractiveness would only occur between the individuals own the same skill factor value. The proposed hybrid strategy uses whether updates the personal best as a trigger to decide the movement model of one particle. In the last chapter, the hybrid strategy increases the particle number that updates their personal best when dealing with a single problem. This helps improve the global best. The MFHA is expected

to increase the proportion of particles that succeed to update personal best by combining the task reassignment and the event-driven strategy. Then MFHA can outperforms MFPSO when processing multiple tasks.

5.3 Benchmark multifactorial problem

Some multifactorial optimization problems are provided to test multitask optimization algorithms' performance. There are 7 commonly used optimization functions. Through rotate, shift and combination of these basic functions, nine multifactorial benchmark problems are presented [12]. For each problem, there are two minimization tasks are included. Considering the intersection and similarity among the tasks in these different problems, where intersection means how many global optimums of the two tasks are identical, includes complete intersection (CI), partial intersection (PI), and no intersection (NI). Within each category of the optimum intersection, there are three categories of similarity based on the Spearman's rank correlation similarity metric of high (HS), medium (MS) and low (LS) similarity. Combine different intersection types and similarity types, 9 problems are provided (CI+HS, CI+MS, CI+LS, PI+HS, PI+MS, PI+LS, NI+HS, NI+MS, NI+LS). It needs to handle these problems as in Eq (5.1) while $K = 2$. Then these multifactorial benchmark problems are used in this test. The benchmark problems are provided in Table 5.2. Then use the multifactorial PSO-FA hybrid algorithm (MFHA) to handle the problems. By comparing it with the multifactorial PSO (MFPSO) to show the hybrid algorithm well perform on multitasking problems.

5.4 Numerical test

In this numerical experiment, there are some parameters provided in Table 6.2. For the PSO model, c_1 and c_2 are set as 1.4. Then, the inertia weight w is set

Table 5.2: Benchmark multifactorial optimization problem

Problem	Task1	Task2
CI+HS	Griewank	Rastrgin
CI+MS	Ackley	Rastrgin
CI+LS	Ackley	Schwefel
PI+HS	Rastrgin	Sphere
PI+MS	Ackley	Rosenbrock
PI+LS	Ackley	Weierstrass
NI+HS	Rosenbrock	Rastrgin
NI+MS	Griewank	Weierstrass
NI+LS	Rastrgin	Schwefel

Table 5.3: Parameter setting

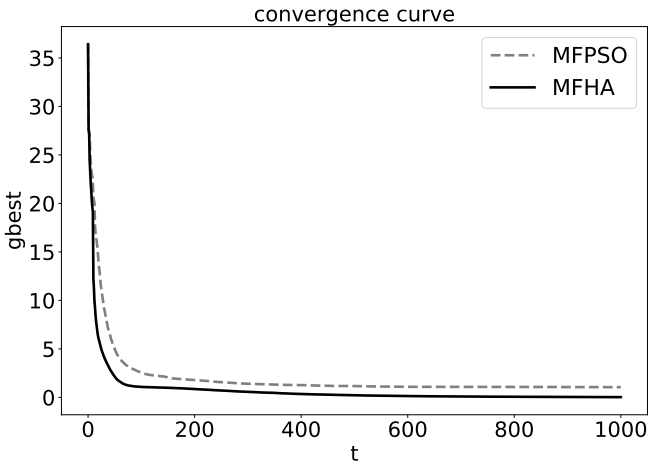
	Parameter Setting
Population size	100
PSO parameter	c_1=1.4, c_2=1.4, w=0.7
r_1, r_2	sample from U(0, 1)
FA parameter	α=1.0, β=1.0, γ=1.0
FA random walk	sample from N(0, 1)
Update time	η=10
Total iteration number	1000
Implement time	20
Dimension	50
Population defination space	[-100, 100]

as 0.7, and r_1 and r_2 independently generated from the uniform distribution over [0,1]. For the FA model, β=1.0, γ=1.0, and the step size parameter α=5. The random walk randomness is sampled from Gaussian distribution with 0 mean and unit variance. The total iteration number is set as 1000. The individual is defined in a 50-dimension space with [-100, 100] space. Each individual is evaluated in each task except for initialization and skill factor reassignment step. Note that the total number of the function call for all tasks is the same in the experiments. This makes the actual iteration number does not strictly equal to 1000. The test will implement for 20 times for each case. The average value of the 20 times run would be recorded for comparison.

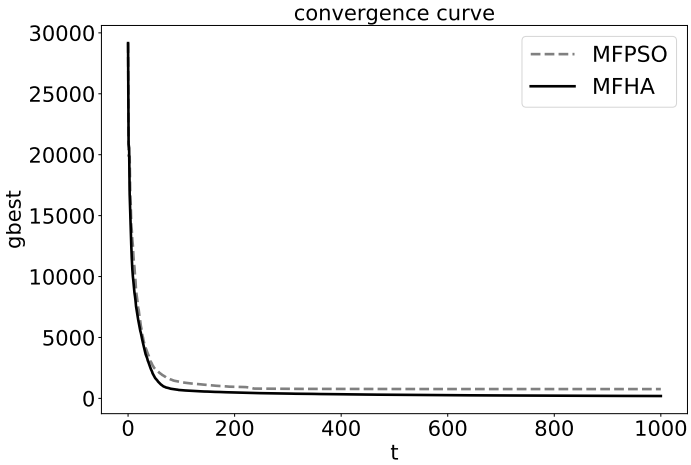
In the test, it compared the multifactorial PSO-FA hybrid algorithm (MFHA)

Table 5.4: Benchmark problem test result (comparison between MFPSO and MFHA, bold values are the better ones)

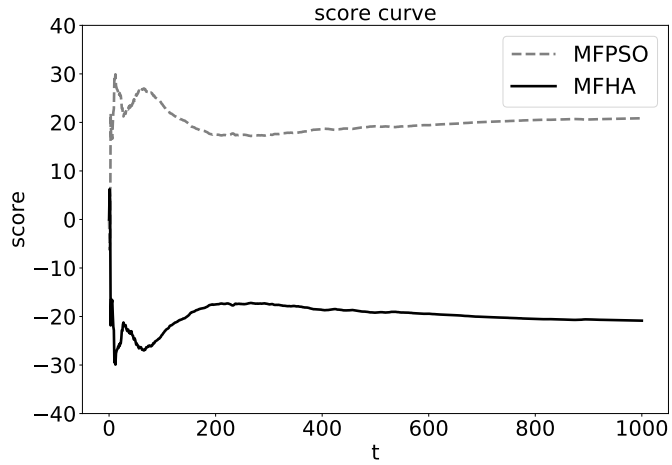
		Task1	Task2	$score_i$
CI+HS	MFPSO	1.0550	762.9851	20.8556
	MFHA	0.0298	193.9934	-20.8556
CI+MS	MFPSO	19.9977	858.0876	30.0333
	MFHA	2.7620	190.5484	-30.0333
CI+LS	MFPSO	20.3607	8011.1641	23.6751
	MFHA	20.1690	7480.6634	-23.6751
PI+HS	MFPSO	1197.8425	2104.4935	19.8088
	MFHA	206.9381	0.7062	-19.8088
PI+MS	MFPSO	19.9987	13451.6652	21.1430
	MFHA	2.5666	296.6262	-21.1430
PI+LS	MFPSO	19.9982	0.5487	1.7809
	MFHA	15.9799	1.0407	-1.7809
NI+HS	MFPSO	12874.4193	885.0240	13.7128
	MFHA	473.8831	194.5595	-13.7128
NI+MS	MFPSO	0.9593	6.8082	8.8262
	MFHA	0.0313	6.7855	-8.8262
NI+LS	MFPSO	1012.5787	8200.8524	13.8607
	MFHA	194.7562	7855.3179	-13.8607



Task1 Griewank

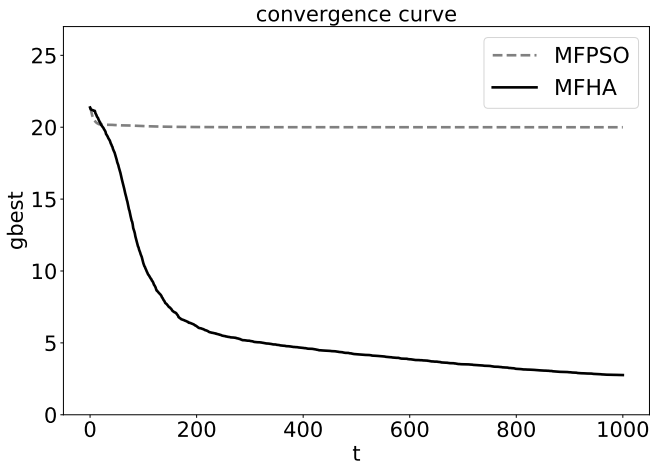


Task2 Rastrigin

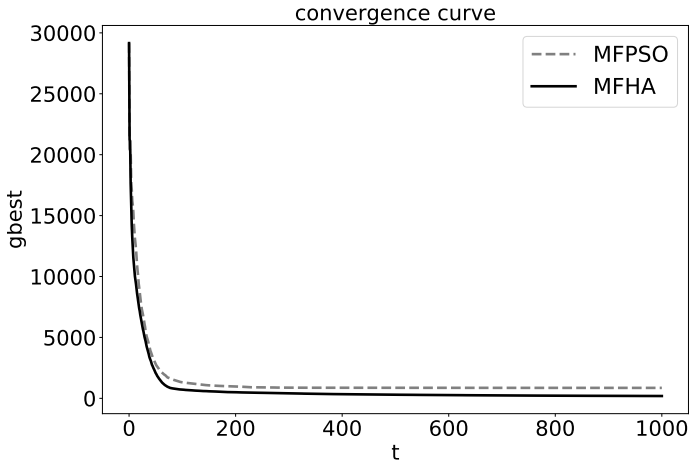


Performance metric trends

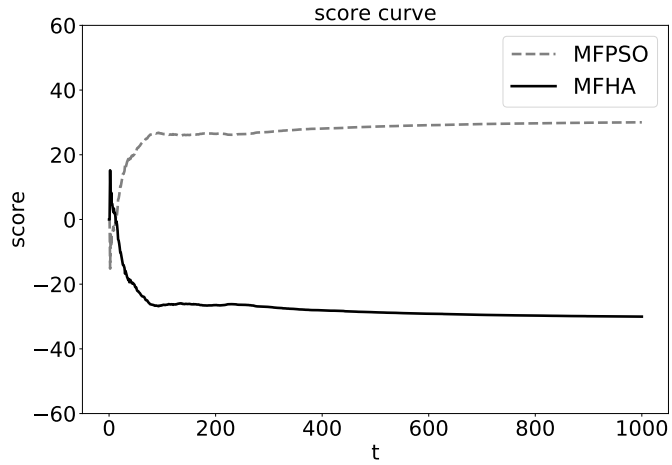
Figure 5.1: CI+HS convergence curve



Task1 Ackley

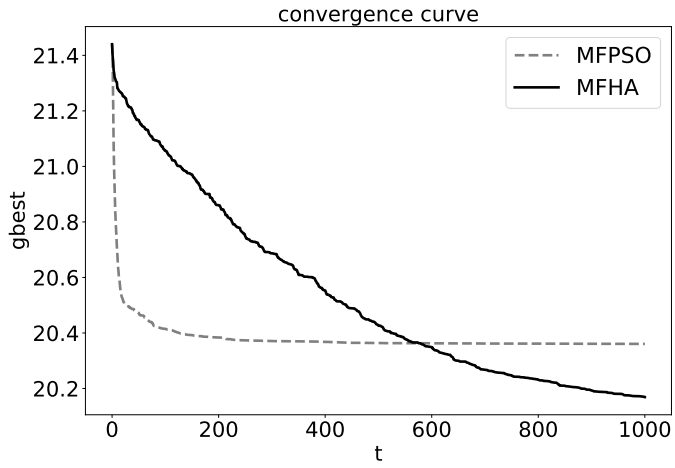


Task2 Rastrigin

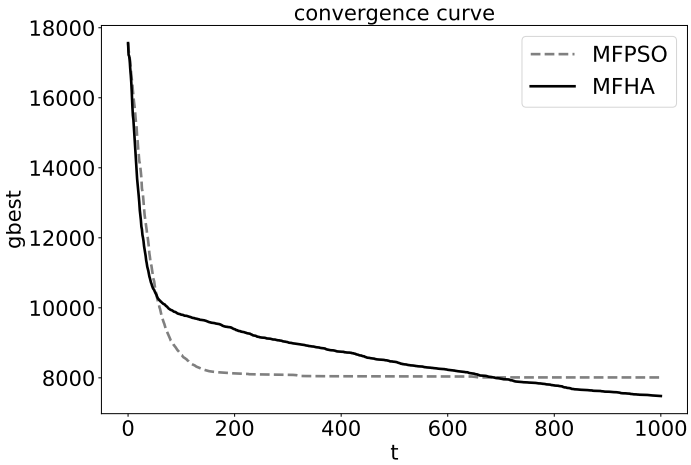


Performance metric trends

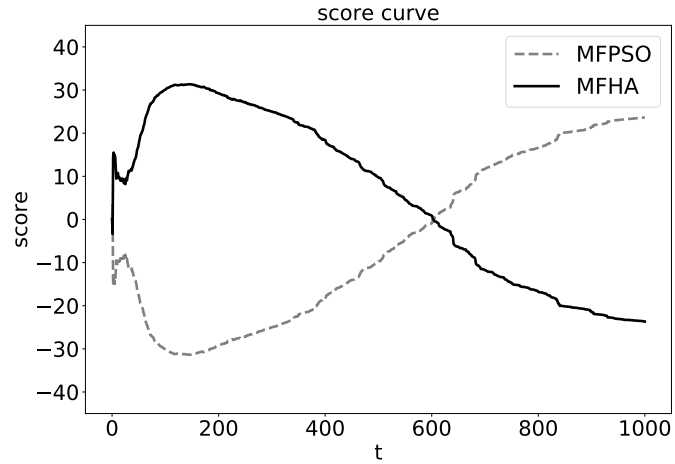
Figure 5.2: CI+MS convergence curve



Task1 Ackley

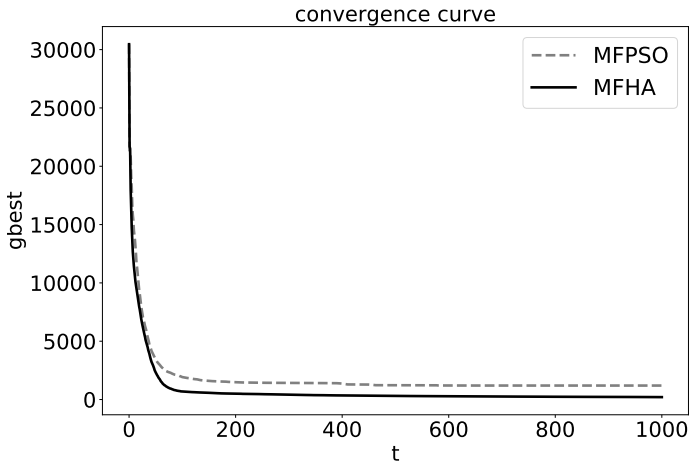


Task2 Schwefel

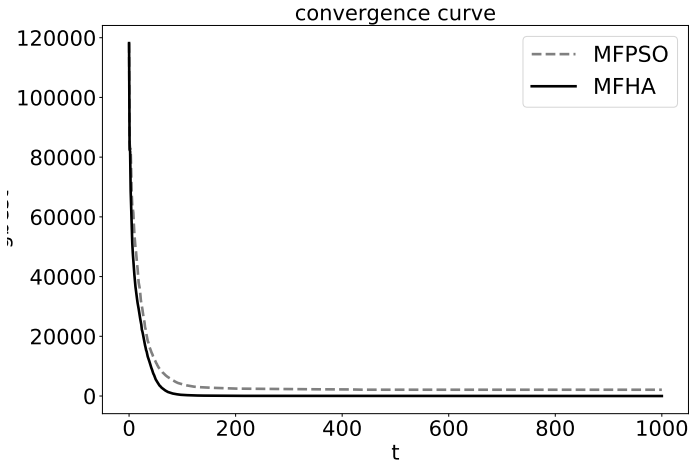


Performance metric trends

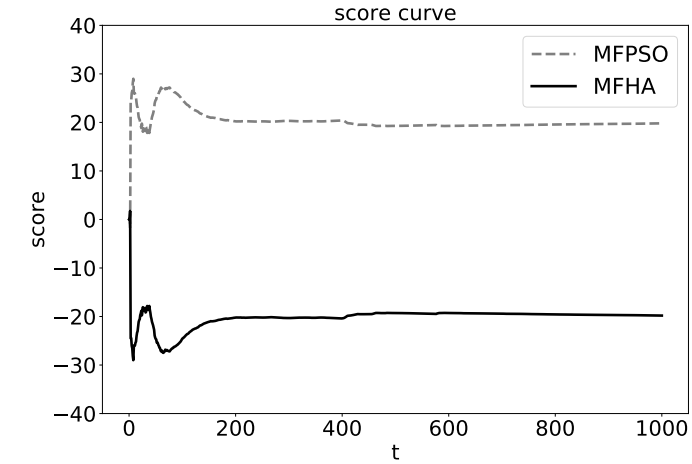
Figure 5.3: CI+LS convergence curve



Task1 Rastrigin

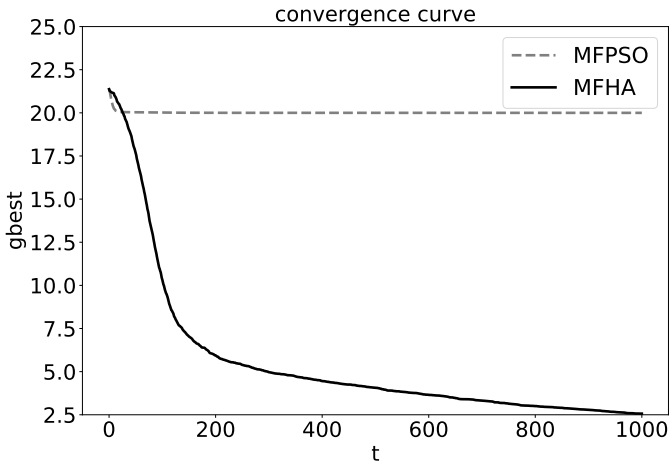


Task2 Sphere

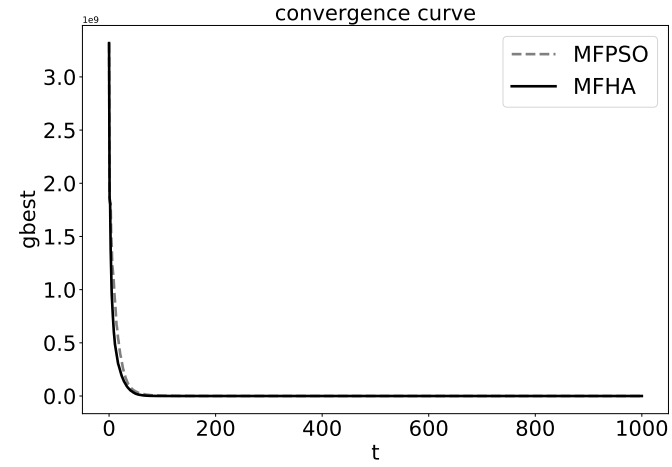


Performance metric trends

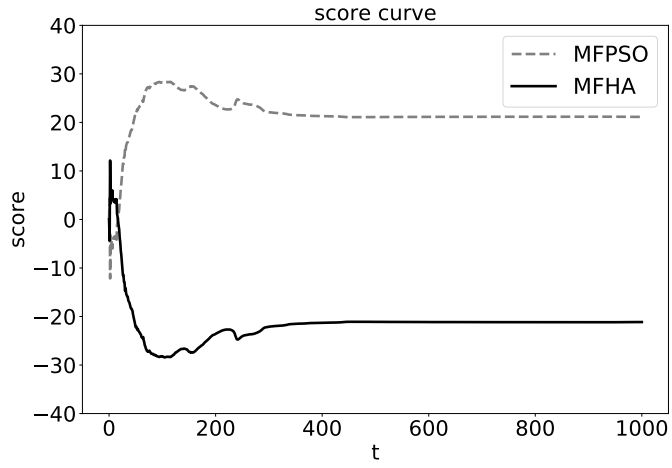
Figure 5.4: PI+HS convergence curve



Task1 Ackley

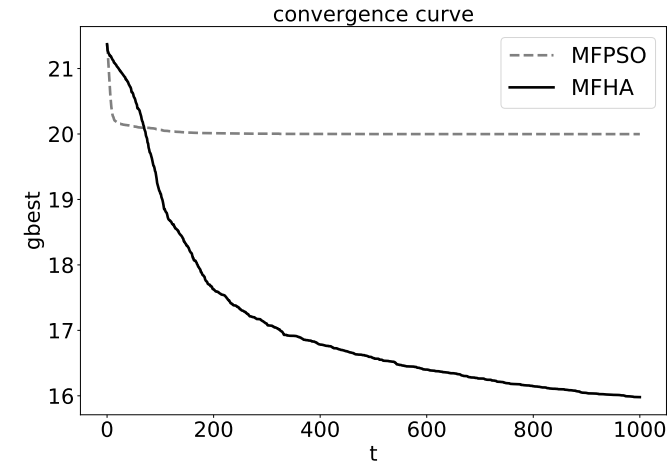


Task2 Rosenbrock

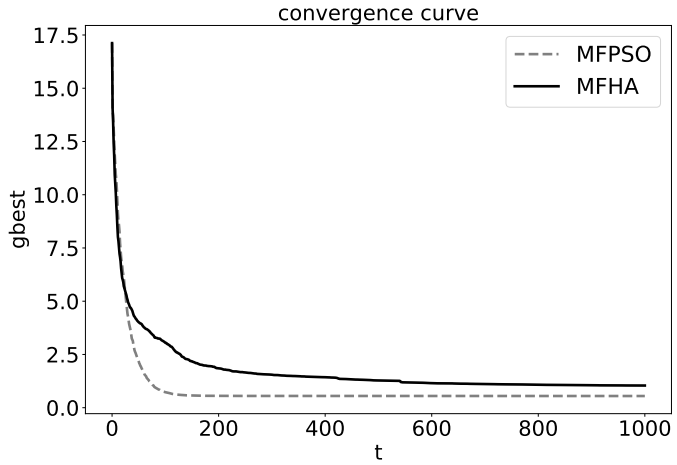


Performance metric trends

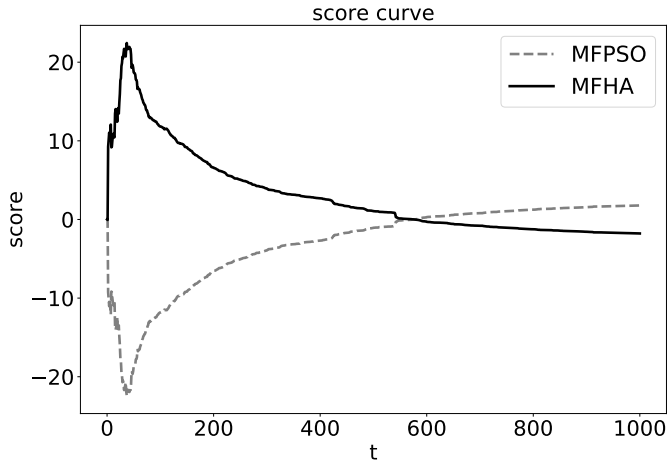
Figure 5.5: PI+MS convergence curve



Task1 Ackley

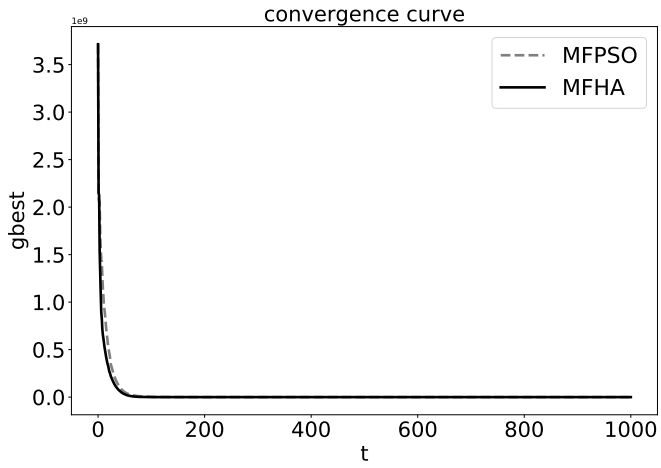


Task2 Weierstrass

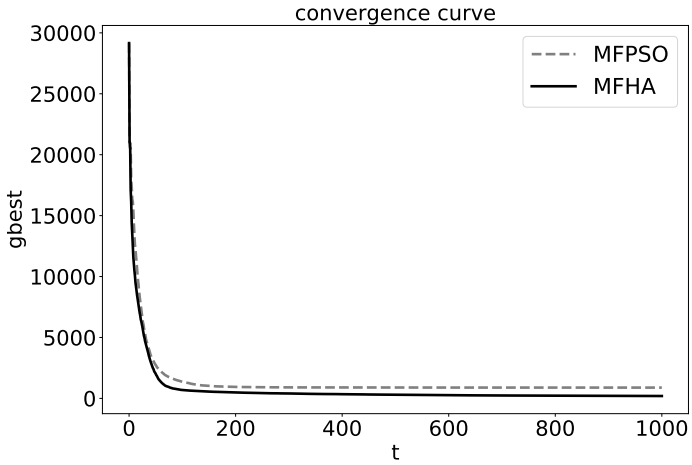


Performance metric trends

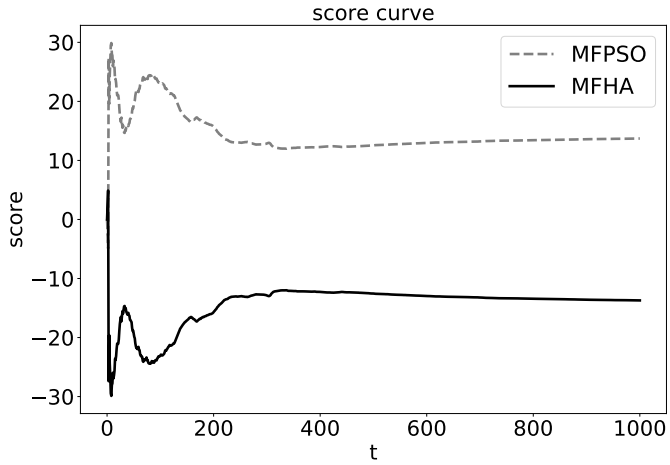
Figure 5.6: PI+LS convergence curve



Task1 Rosenbrock

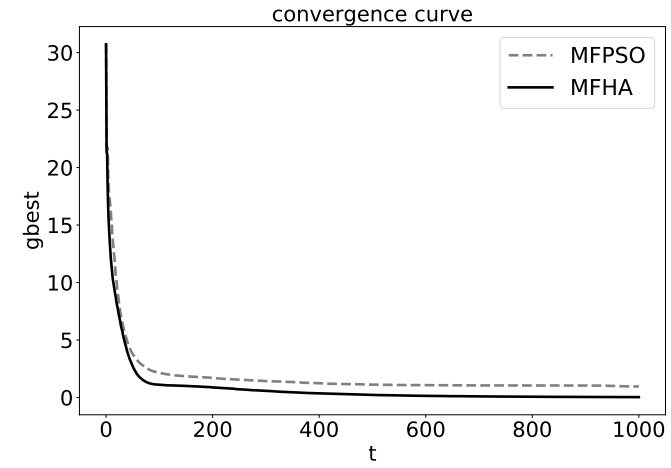


Task2 Rastrigin

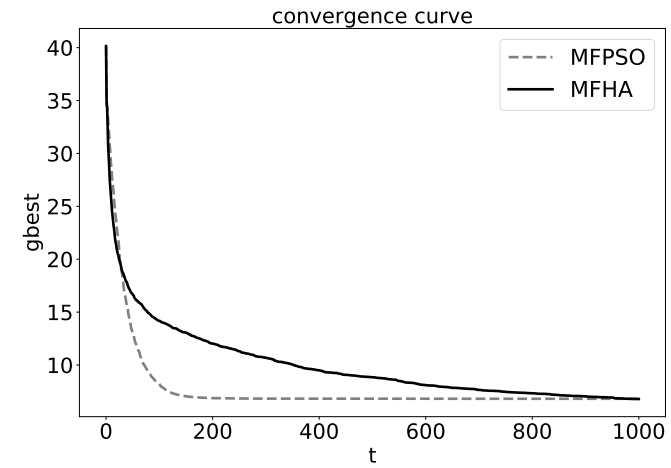


Performance metric trends

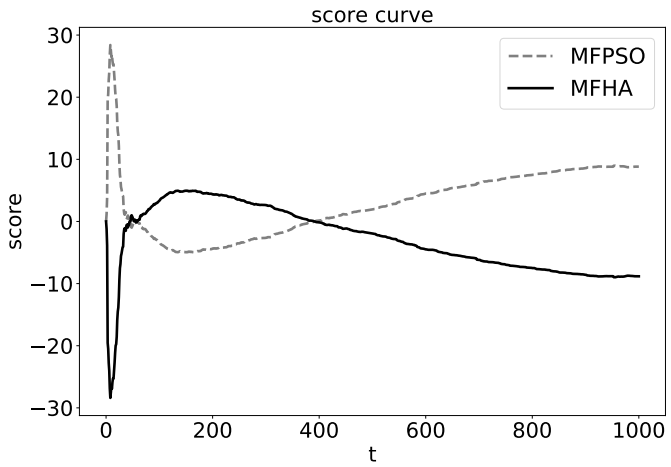
Figure 5.7: NI+HS convergence curve



Task1 Griewank

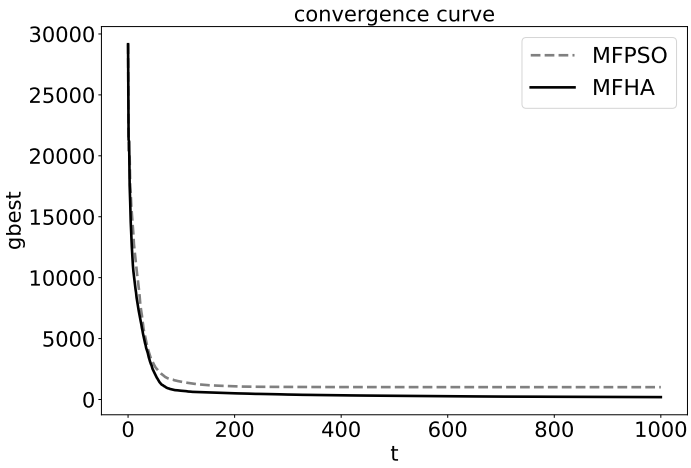


Task2 Weierstrass

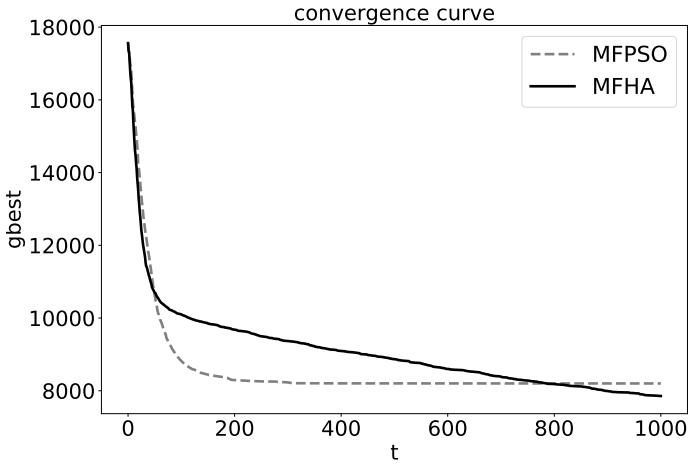


Performance metric trends

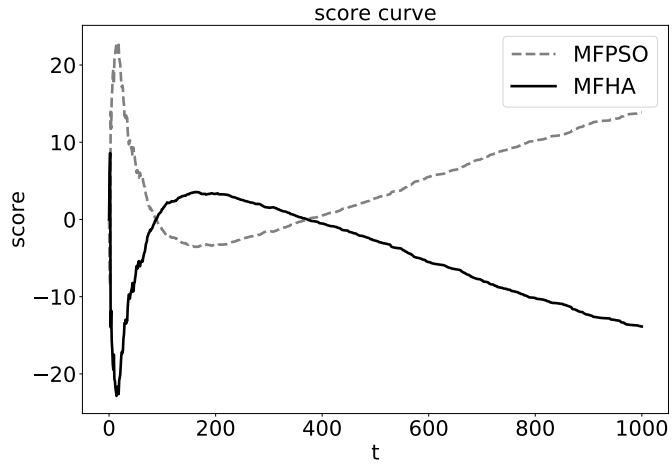
Figure 5.8: NI+MS convergence curve



Task1 Rastrigin



Task2 Schwefel



Performance metric trends

Figure 5.9: NI+LS convergence curve

with multifactorial PSO (MFPSO). In Figure 5.1 - Figure 5.9, the horizontal axis means the iteration numbers while the vertical axis means function values of the best memory record. These figures describe the update situation of the global best information in each algorithm.

In order to compare the performance of different algorithms, a simple performance metric is used from the paper [12]. For 2 algorithms, $A_a(a = 1, 2)$ describe them. In one problem, 2 minimization tasks $T_k(k = 1, 2)$ exist. Each algorithm is run for L repetitions while $I(a, k)_l$ denotes the best result obtained in the l-th repetition by Algorithm A_a on the task k. While μ_k and σ_k describe the mean and the standard deviation for task T_k over all the repetitions of the algorithms. Then the normalized performance $I'(:, k)_l = (I(:, k)_l - \mu_k) = \sigma_k$ is considered for all tasks. For each algorithm A_a, its performance score is given as in Eq (5.4).

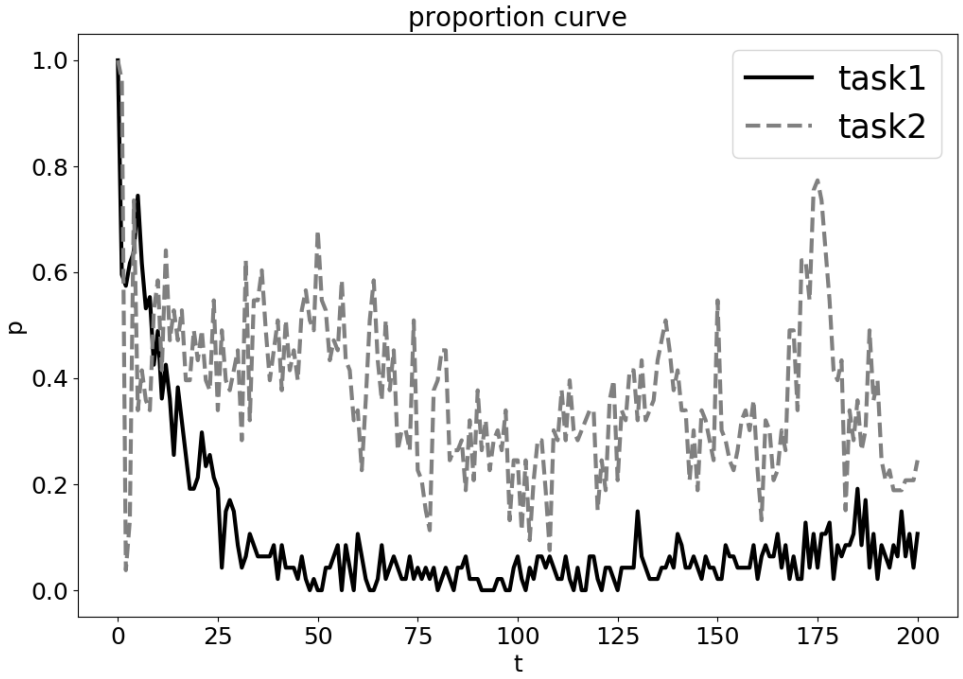
$$\text{score}_a = \sum_{k=1}^{K} \sum_{l=1}^{L} I'(a, k)_l \tag{5.4}$$

The test result of comparing MFPSO and MFHA is shown in Table 5.4 and Figure 5.1 - Figure 5.9. Here used η=10 to reassign the skill factor for the multifactorial algorithms. The average result and final score are shown in Table 5.4. It records the average result of each task and the final score in one problem. The smaller score value means that the algorithm performs better in the problem. From the test result, MFHA gets the best score in whole problems than MFPSO. Even though MFHA loses to MFPSO in TASK1 of CI+LS, NI+MS, the hybrid swarm still gets better scores than MFPSO. That means the MFHA gets better optimums than MFPSO averagely for the multifactorial optimization benchmark.

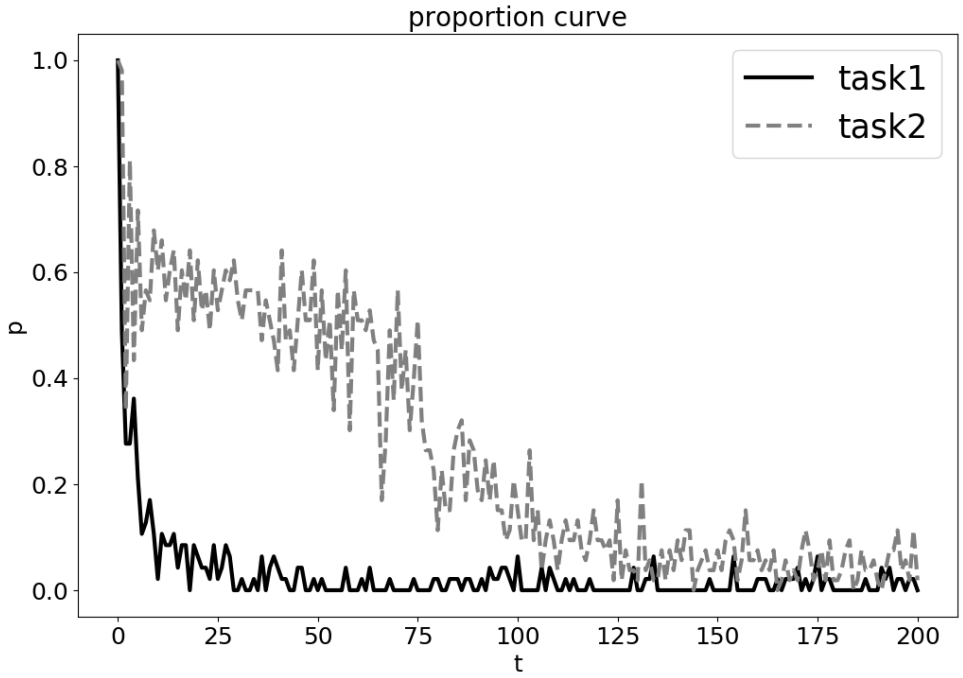
From the result, it is able to observe that in all high similarity problems, both MFPSO and MFHA perform a high improvement rate. For CI+MS, MFHA loses to MFPSO in the start stage in one task but keep progress trend to exceed MFPSO when MFPSO stop improvement quickly. For CI+LS, MFHA loses to MFPSO in the start stage in both tasks and exceed MFPSO in one task when MFPSO stop

improve quickly. The score result shows that MFHA performs better than MFPSO in whole cases.

The proposed method helps the particle to do the local search if it updates its personal best. Combining the proposed strategy with the multifactorial optimization and expects the population to increase the particles that update their personal best. Here show some examples of one run's result with 200 iterations of particle number record of CI+MS problem in Figure 5.10, 5.11. In these figures, the vertical axis describes the proportion of the particles that update their personal best in two tasks. The horizontal axis records iteration numbers. When tasks reassignment is not available, two single particle group that does not change members will process one of the tasks in MFPSO, MFHA. From the result, it can see that compare to task2, less particles update their personal best in MFPSO and MFHA. Then in the figures that task reassignment is available, MFPSO increases the proportion of particles that update their personal best at 10 iteration since the task reassignment time is 10. But MFPSO not keeps the high proportion value for a long time. Compare to MFPSO, MFHA also increases the proportion value at 10 iteration and keeps a high proportion values for a while. That indicates that the cooperation of the event-driven hybridization and skill factor reassignment improves the particles that update their personal best when simultaneously processing multiple tasks.

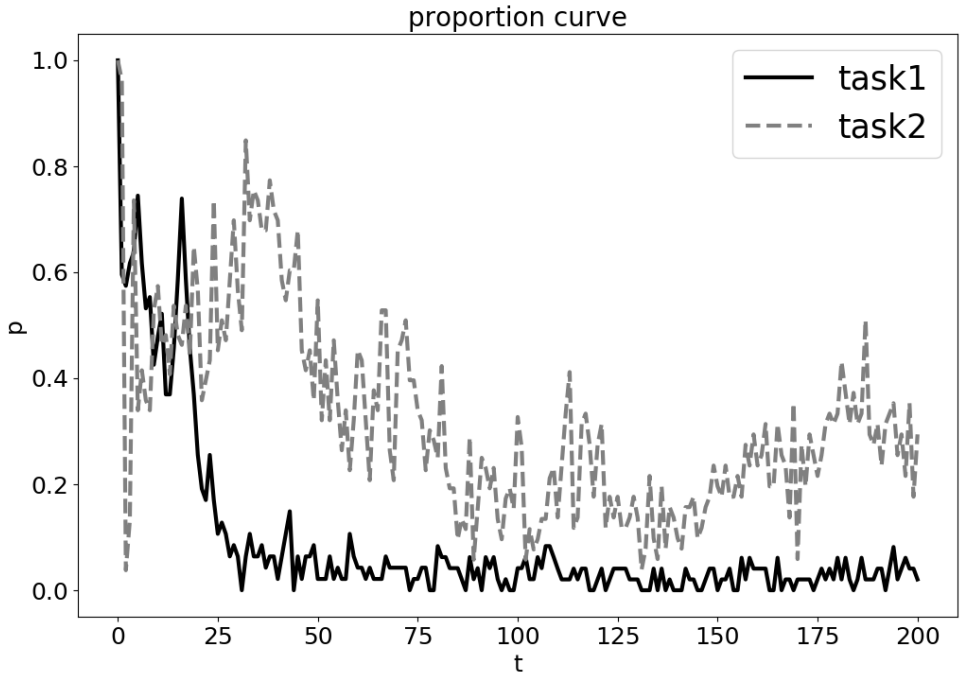


Proportion of particles that update personal best in MFPSO

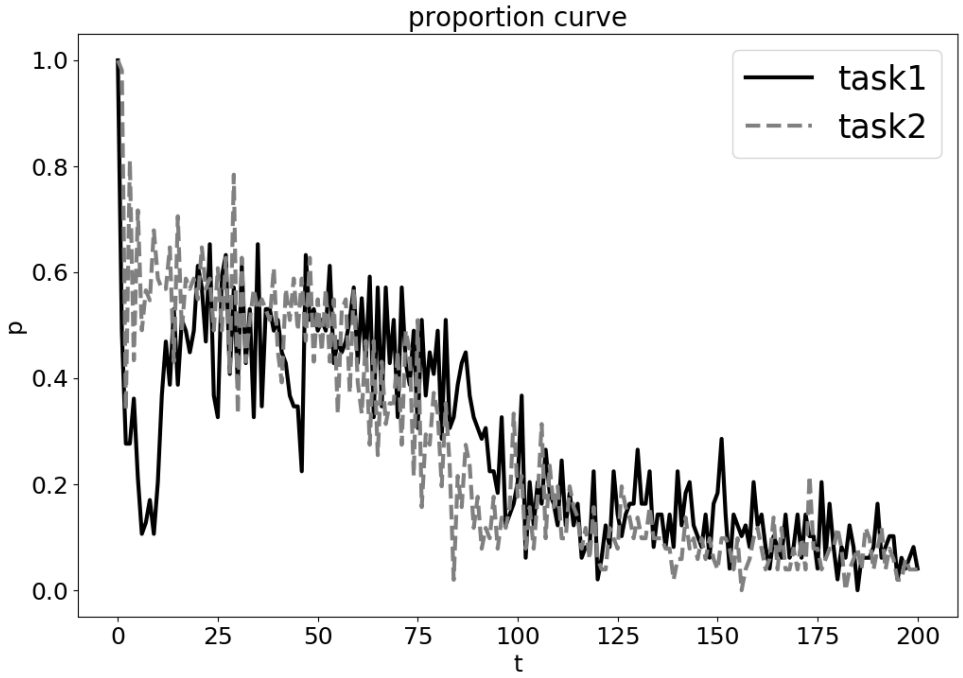


Proportion of particles that update personal best in MFHA

Figure 5.10: One run's result of 200 iterations of no task reassignment in CI+MS problem



Proportion of particles that update personal best in MFPSO



Proportion of particles that update personal best in MFHA

Figure 5.11: One run's result of 200 iterations of task reassignment in CI+MS problem

Chapter 6

Real-world application

It is important to improve efficiency, reduce costs, develop new products in some fields as manufacturing. These are usually provided as optimization problems. These optimization problems are usually categorized as single-objective or multi-objective, linear or nonlinear, convex or nonconvex, and discrete or continuous. Among these actual problems, a lot of them are black-box optimization problems for that the problems are difficult to describe. Thus, some optimization solvers are used to processing such problems with automatic search behavior. Among them, metaheuristics [62] are attractive methodologies since they require only algorithmic knowledge rather than problem domain-specific knowledge.

Swarm intelligence [29] is a typical metaheuristic for black-box function optimization. By making hybridization of selected swarm intelligence models, it is able to utilize the different properties of them. On another side, multitask optimization has been proposed as a novel paradigm in evolutionary computation recently for that it utilizes parallelism of a population-based search mechanism. It will be significant to research multitask optimization problems when some intersections existed among different tasks. Multifactorial optimization is provided for processing problems like this. Then, inspired by the intelligent behavior of human's ability to deal with multiple tasks, the multifactorial evolutionary algorithm (MFEA) [12] is provided for the multitask optimization problems. MFEA is

a method with the evolutionary process making use of evolutionary multitasking to enable the autonomous transfer of knowledge from one problem to the other problem. As shown in the previous research, it is important to decide how the solver deals with different tasks at the same time. In MFEA, a skill factor is used to automatically assign tasks for a population. With the skill factor, individuals would focus on themselves preferred tasks and only computing them. By importing the task selection method into a population-based algorithm, it is able to make the solver simultaneously processing multiple problems.

For some real-world problems, some optimization tasks are complex for they have many constraints. The multiple car design benchmark [31] has been proposed as an engineering design optimization. By applying a multifactorial hybrid swarm intelligence algorithm and use it to process the benchmark problem, it is able to verify whether the multifactorial hybrid algorithm can simultaneously process multiple cars design tasks well. In this chapter, the multifactorial PSO-FA hybrid algorithm is used. By testing it on the multiple car design benchmark and comparing it with multifactorial PSO, then check whether the proposed hybrid algorithm improved than an algorithm without the hybrid scheme [55].

6.1 Real-world multiple car design problem

In the real problem, it costs too much computation and not easy to be solved. Considering the limited computation budget, the benchmark problem is used instead of the real problem. This section introduces the Mazda multi-objective benchmark optimization problem for the numerical test. This benchmark is designed for testing the performance of search-based optimization methods such as evolutionary algorithms and swarm intelligence. Two optimization problems are included in this benchmark, one is a single-objective optimization of weight minimization of multiple cars while another is bi-objective optimization of weight and number of common parts in these cars.

There are three target cars that are different types as compact, medium, and wagon. The simultaneous design of multiple cars is expected to reduce development cost and increasing common parts that may help reducing the manufacturing cost. When handling the actual engineering design problem, constraints conditions are rather important as engineering design, body stiffness, safety for impact, and so on. In the previous research [16, 35], different methods are proposed for processing single-objective or multi-objective problem in this benchmark. This work aims to provide a hybrid algorithm that well performs to simultaneously optimize the mass of the three cars.

In the Mazda multiple car design benchmark problem, it needs to get the minimum mass of three different kinds of cars that are SUV, CDW, and C5H. These three cars own different sizes and body shapes, but the same number of parts. In one car, each part relates to the corresponding car weight and strength to crush. This means that each car must satisfy the standard body strength.

In practice, there are 222 design parameters provided in the benchmark problem where each car has a set of 74 design parameters and each design parameter set can be separable among different cars. These design parameters would be used to calculate the mass of one car and their strength is able evaluated by numerical simulation. When handling the problem, three cars can be simultaneously optimized as three different optimization tasks. It is able to treat the benchmark problem as a typical multitasking problem for that it expects to utilize some common design knowledge exist among different cars.

The original design parameters in dividing each car are denoted by, $x_{SUV} = (x_1, x_2, ..., x_D)$, $x_{CDW} = (x_{1+D}, x_{2+D}, ..., x_{2D})$, $x_{C5H} = (x_{1+2D}, x_{2+2D}, ..., x_{3D})$ for each car, and $D = 74$ in this problem.

Design parameters are like (6.1)–(6.3).

$$x_{SUV} = (x_1, x_2, ..., x_d), d = 74 \tag{6.1}$$

$$x_{CDW} = (x_{1+d}, x_{2+d}, ..., x_{2d}) \tag{6.2}$$

$$x_{SUV} = (x_{1+2d}, x_{2+2d}, ..., x_{3d}) \tag{6.3}$$

As a multifactorial optimization problem, each individual is represented by $x_j \in \mathbb{R}^D$, and j denotes an index of the individual. A skill factor is able to assign each individual to the corresponding task, that is if a skill factor $\tau_i = k$ then x_j is treated as x_k. Here k, k = 1,2,3 represent car types, that is $k = 1$ means car label is SUV, 2 means CDW, and 3 means C5H. For each car, the same dimension owns common upper limitation and lower limitation as follows,

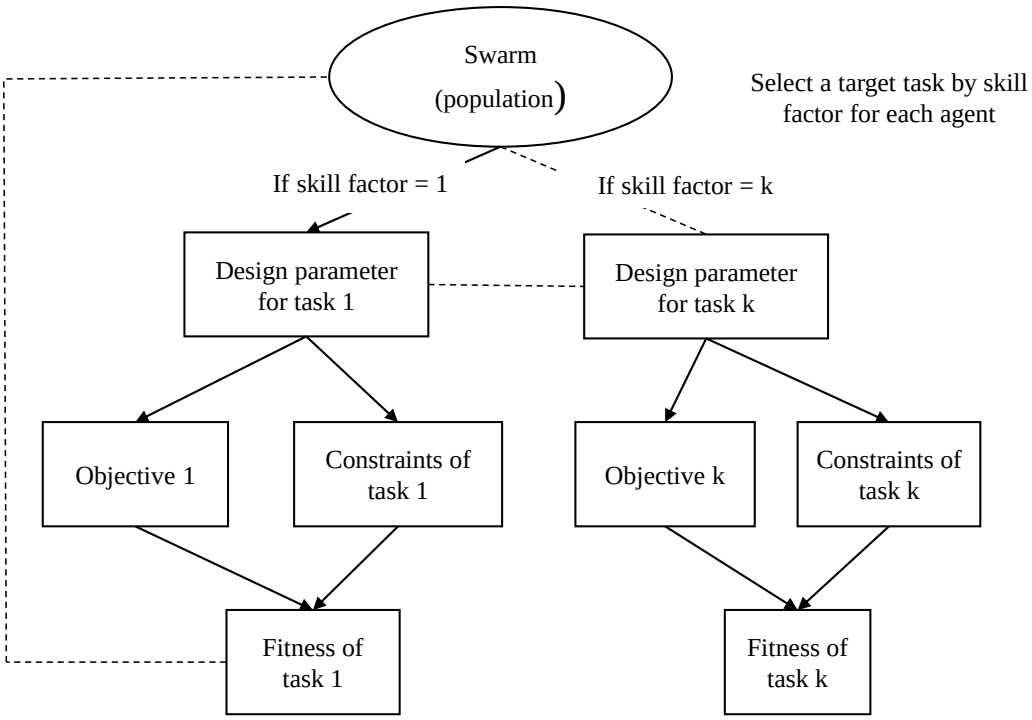


Figure 6.1: Multiple car design problem

$$L_d^k \leq x_d^k \leq U_d^k, k = 1, 2, 3 \quad d = 1, 2, ..., 74 \tag{6.4}$$

There are 18 constraint conditions concerning with car body strength and safety for impact, and so on, these are denoted by the functions,

Table 6.1: Performance evaluation item

Performance evaluation item	related constraint
Full frontal impact	g_1^k
Frontal offset impact	g_2^k, g_3^k, g_4^k
Side impact	g_5^k, g_6^k, g_7^k
Rear impact	g_8^k, g_9^k
Low eigen value	$g_{10}^k, g_{11}^k, g_{12}^k$
Body stiffness	g_{13}^k, g_{14}^k

$$g_i^k \geq 0, k = 1, 2, 3 \quad i = 1, 2, ..., 18 \tag{6.5}$$

Here $i; i = 1; 2; \cdots ; 18$ indicate the constraint index. The contents of the 14 constraints is shown in Table 6.1. These constraints concern with crashworthiness of each car. They are modeled by a huge number of simulations in supercomputer and the Kriging method. The remaining constraint $g_{15}; \cdots ; g_{18}$ of each car were defined by the following 4 relationships,

$$\begin{aligned} g_{15}^k &= x_{14}^k - x_{13}^k \\ g_{16}^k &= x_{16}^k - x_{15}^k \\ g_{17}^k &= x_{14}^k - x_{64}^k \\ g_{18}^k &= x_{15}^k - x_{64}^k \end{aligned}$$

For processing multiple constrained multitasking problems, it needs to handle multiple constraints simultaneously [47]. The method is to integrate multiple constraint violations into a single violation measure. Generally, the sum of its constraint violations, the number of the violated constraints or the violation of the most violated constraint are used in this kind of problem.

Then set a penalty mechanism for this problem. For these all constraints need to be none negative, so add all negative ones with an impact issue to each car's mass computation result as the evaluation value.

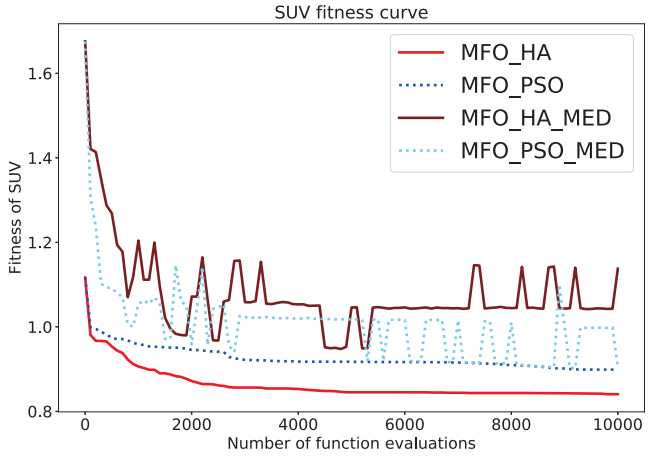
$$fitness^k = mass^k + \rho * penalty^k, k = 1, 2, 3 \tag{6.6}$$

6.2 Numerical test

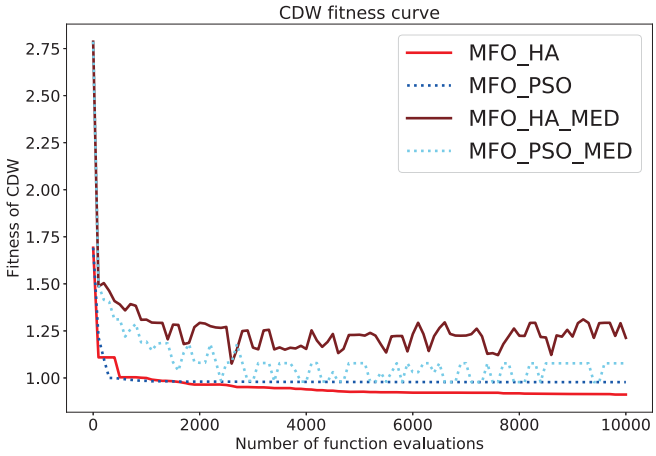
About the test conditions, individual number is set as 300 (at the initial step there are 200 particles and 100 fireflies), and a totally 30000 times evaluation is permitted. The parameter setting of PSO and FA is summarized in Table 6.2. Then the weight in the fitness function is set as $\rho = 1$. First, an interval of a skill factor update is examined. Here an interval $\eta = 1$ means the skill factor would be updated at every step. The obtained results based on the test where $\eta = 10$. As a result, it selects 10 steps as the interval of a skill factor update. Then, the test compared the performance between the multifactorial PSO-FA hybrid algorithm and multifactorial PSO. The obtained best mass record in each algorithm is shown in Table 6.3. For all cars, the proposed hybrid algorithm achieved better mass records than PSO based method.

From the test result, the multifactorial PSO-FA hybrid algorithm outperforms in optimizing three minimum optimization problems simultaneously than the multifactorial PSO. The fitness and the obtained best mass are shown in Figure 6.2 and Figure 6.3, respectively. Here the obtained best mass means that it achieved the light body design in this benchmark within the alternative that satisfied all constraints. The difference between fitness and mass corresponds to the number of constraint violations. At the early stage, it can find that the best solution achieves all constraints satisfaction. Then, the proposed hybrid algorithm can keep the diversity of swarm, as shown in Figure 6.2. Also, it can locally search the boundary region of constraints by FA based operation and improve the performance in each optimization task.

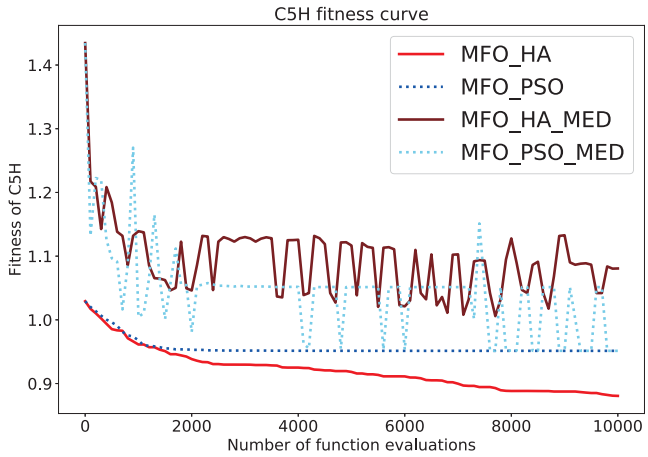
In the previous study, it has presented the multifactorial PSO by introducing a multifactorial mechanism into a simple PSO. Then extending a simple multifactorial PSO to PSO-FA hybrid algorithm, and expect a hybridization approach would improve the performance on multifactorial optimization problems as on single-objective optimization problems. In this chapter, here introduced a multi-



SUV

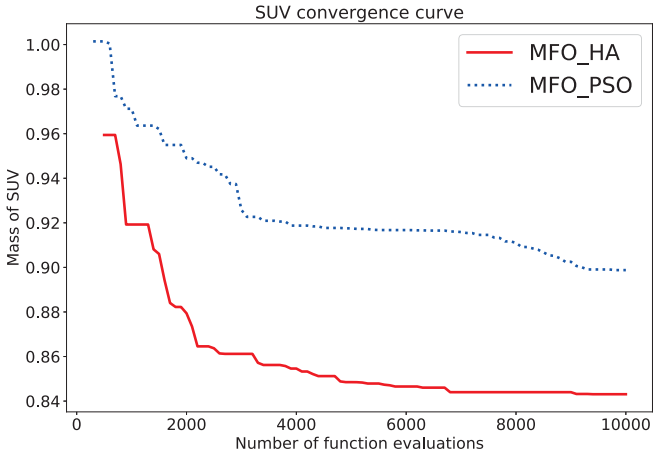


CDW

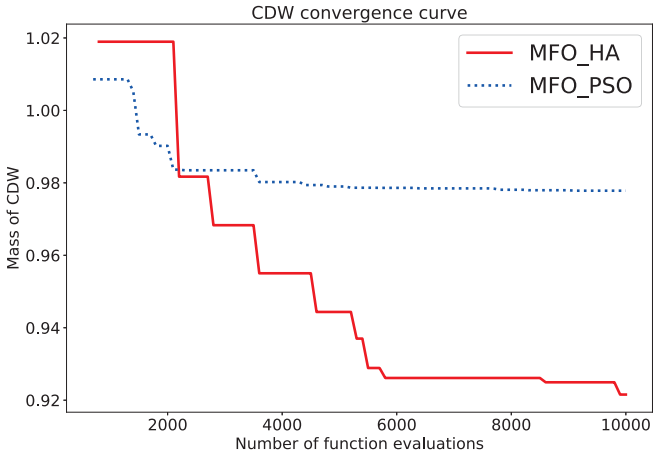


C5H

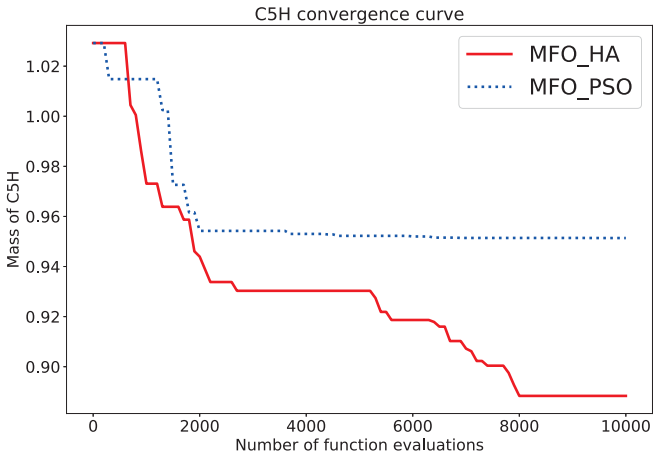
Figure 6.2: The fitness value trajectories of the global best memory and median in the swarm for each car. Upper two lines, brown and cyan indicate the median fitness values of the proposed PSO-FA hybrid multifactorial algorithm and PSO based multifactorial algorithm, respectively.



SUV



CDW



C5H

Figure 6.3: The best mass value trajectories for each car.

Table 6.2: Parameter setting

Population size	300
PSO parameter	w=0.7
	c_1=1.4, c_2=1.4
	r_1, $r_2 \sim$ U(0, 1)
FA parameter	α=0.02
	β=1.0
	γ=0.6

Table 6.3: optimum comparison

	SUV	CDW	C5H
MFO PSO	0.9056	0.9568	0.9352
MFO PSO-FA hybrid algorithm	0.8319	0.9185	0.8818

factorial PSO-FA hybrid algorithm where each agent accords to either PSO or FA model with sharing the personal best memory. In this hybrid algorithm, the agent will change its model if it could update the global best memory and his original model is FA. This property helps local search around the global best memory. Then, a task selection by the skill factor makes the population into as like island model that corresponds to the assigned task. In other words, an individual would be assigned a skill factor to decide which task it good at.

I have examined the proposed multifactorial PSO-FA hybrid algorithm to Mazda multiple car design benchmark problem and compared its performance with a simple multifactorial PSO. I design a mechanism that makes each agent that is representing the design variables gets close to its own leader by assigned task, and task reassignment is carried out at regular intervals in the proposed algorithm. In addition, each agent would affect much by its leader in the proposed hybrid algorithm since it utilizes the global best in the PSO model and the best agent in the Firefly model. In the result of the numerical test, it performed better convergence trend in the multifactorial PSO-FA hybrid algorithm. This result shows that hybridization helps to improve PSO about multifactorial optimization problem.

Chapter 7

Conclusion

The optimization problem is always challenging in many fields. Among the optimization problems, the black-box optimization problem is difficult for the unknown landscape. For black-box optimization problems, metaheuristics are used for that they can process problems with incomplete information.

As a population-based metaheuristic, swarm intelligence is inspired by natural collective behavior. By simulating the personal behavior and social communication of the biological systems, many swarm intelligence algorithms are developed. As one swarm intelligence algorithm, PSO has an easy implementation for the optimization problem. Even though PSO has a fast convergence speed, the poor local search limits the evolution of PSO. Since FA has a strong local search and does not be affected by the global best, this paper considers how to make a better solver by developing a hybrid algorithm of PSO and FA.

Previous works using parameter tuning, model modifying to build a hybrid. Parameter tuning costs a lot of trial and error. However, high-dimensional problems own a lot of parameters that need to be evaluated. Parameter tuning would be expensive for computation. Adjusting particle number with fixed parameters can also change the performance of one hybrid algorithm.

This research proposes making a hybrid swarm of standard models without parameter tuning. By utilizing an event-driven trigger to help particles to get

rid of the global best. By utilizing an event-driven trigger based on whether the personal best information is updated, it is able to control one particle moves according to PSO model or FA model. CEC 2015, CEC 2017 provide some expensive benchmark problems with known properties. To show how the hybrid algorithm performs well, test the hybrid algorithm on the benchmark optimization problems, and compare it with PSO, FA, SPSO, HFPSO. From the numerical test, it shows that the progress is improved by the proposed hybrid when comparing it with others. For the problems that own a lot of local optima, the proposed hybrid improves the global best than others.

Except for handling a single problem, this research applies the proposed hybrid to multifactorial optimization to simultaneously process multiple optimization tasks with a single population. There may exist some relationship among these tasks. With a task reassignment mechanism, it would give particles a chance to change their preferred tasks. The hybrid strategy is designed to increase the particles that update their personal best. Combining the proposed strategy with multifactorial optimization is expected to help the proportion of the particles when processing multiple tasks. From the test result in Chapter 5, the proposed hybrid outperforms multifactorial PSO. Even for the tasks that are not similar to each other, the proposed hybrid wins from a total evaluation. The combination of hybridization strategy and multifactorial optimization contributes to improving the performance of simultaneously handling multiple optimization tasks.

In chapter 6, a black-box benchmark of a real problem in which it needs to optimize the mass of three different cars is provided. Then using the multifactorial PSO-FA hybrid algorithm to process this problem and comparing the hybrid swarm and the multifactorial PSO to show that the multifactorial hybrid swarm success to show well performance on a real problem.

This research contributes to the inspiration of how to make a better algorithm by making a hybrid of PSO and FA. The event-driven PSO-FA hybrid is proposed and show good performance in high-dimension benchmark problems. By combing

it with multifactorial optimization, the algorithm improves the best solutions in multiple tasks. Applying it to a benchmark black-box car design problem which is near to a real problem to show its good performance for a black-box problem. For future works, I consider to find a better method to control the personal study process and solving more practical problems.

Acknowledgement

I feel grateful for Professor Suzuki, Professor Morita, Professor Fujisaki for the kindly help and advice. They give important comments to help me complete this thesis. And I am grateful to Mr. Hatanaka for advice and support to this research. They are generous and shared lots of expensive experiences with me. Also thankful to whole Suzuki Lab members for their cooperation, friendship and encouragement.

And presented papers in this thesis were supported by JPSS Program for Leading Graduate Schools and this work is partially supported by "Program for Leading Graduate Schools" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] Nor Azliana Abdullah, Nasrudin Abd Rahim, Chin Kim Gan, and Noriah Nor Adzman. 2019. Forecasting Solar Power Using Hybrid Firefly and Particle Swarm Optimization (HFPSO) for Optimizing the Parameters in a Wavelet Transform-Adaptive Neuro Fuzzy Inference System (WT-ANFIS). *Applied Sciences* 9, 16 (2019), 3214.

[2] Moyinoluwa B Agbaje, Absalom E Ezugwu, and Rosanne Els. 2019. Automatic data clustering using hybrid firefly particle swarm optimization algorithm. *IEEE Access* 7 (2019), 184963–184984.

[3] Ammar AQ Ahmed and D Maheswari. 2017. Churn prediction on huge telecom data using hybrid firefly based classification. *Egyptian Informatics Journal* 18, 3 (2017), 215–220.

[4] Hazem Ahmed and Janice Glasgow. 2012. Swarm intelligence: concepts, models and applications. *School Of Computing, Queens University Technical Report* (2012).

[5] Niam Abdulmunim Al-Thanoon, Omar Saber Qasim, and Zakariya Yahya Algamal. 2019. A new hybrid firefly algorithm and particle swarm optimization for tuning parameter estimation in penalized support vector machine with application in chemometrics. *Chemometrics and Intelligent Laboratory Systems* 184 (2019), 142–152.

[6] S Arunachalam, T AgnesBhomila, and M Ramesh Babu. 2014. Hybrid particle swarm optimization algorithm and firefly algorithm based combined economic and emission dispatch including valve point effect. In *International Conference on Swarm, Evolutionary, and Memetic Computing*. Springer, 647–660.

[7] Ibrahim Berkan Aydilek. 2018. A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. *Applied Soft Computing* 66 (2018), 232–249.

[8] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz. 1997. *Handbook of Evolutionary Computation*. Oxford University Press.

[9] Vaibhav Bhargava, Seif-Eddeen K Fateen, and Adrian Bonilla-Petriciolet. 2013. Cuckoo search: a new nature-inspired optimization method for phase equilibrium calculations. *Fluid Phase Equilibria* 337 (2013), 191–200.

[10] Qiong Chen, B Liu, Qiang Zhang, J Liang, Ponnuthurai Suganthan, and B Qu. 2014. Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization. *Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University* (2014).

[11] Qunjian Chen, Xiaoliang Ma, Yiwen Sun, and Zexuan Zhu. 2017. Adaptive memetic algorithm based evolutionary multi-tasking single-objective optimization. In *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 462–472.

[12] Bingshui Da, Yew-Soon Ong, Liang Feng, A Kai Qin, Abhishek Gupta, Zexuan Zhu, Chuan-Kang Ting, Ke Tang, and Xin Yao. 2017. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems,

performance metric, and baseline results. *arXiv preprint arXiv:1706.03470* (2017).

[13] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1, 1 (2011), 3–18.

[14] A Francis Saviour Devaraj, Mohamed Elhoseny, S Dhanasekaran, E Laxmi Lydia, and K Shankar. 2020. Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization algorithm for energy efficient load balancing in Cloud Computing environments. *J. Parallel and Distrib. Comput.* (2020).

[15] Mohammed El-Abd. 2011. A hybrid ABC-SPSO algorithm for continuous function optimization. In *2011 IEEE Symposium on Swarm Intelligence*. IEEE, 1–6.

[16] Hiroaki Fukumoto and Akira Oyama. 2018. Benchmarking multiobjective evolutionary algorithms and constraint handling techniques on a real-world car structure design optimization benchmark problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 177–178.

[17] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. 2009. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing* 13, 10 (2009), 959.

[18] Mounir Ben Ghalia. 2008. Particle swarm optimization with an improved exploration-exploitation balance. In *2008 51st Midwest Symposium on Circuits and Systems*. 759–762.

[19] Amirhossein Ghodrati and Shahriar Lotfi. 2012. A hybrid cs/ga algorithm for global optimization. In *Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011*. Springer, 397–404.

[20] Mahya Mohammadi Golchi, Shideh Saraeian, and Mehrnoosh Heydari. 2019. A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation. *Computer Networks* 162 (2019), 106860.

[21] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. 2015. Multifactorial evolution: toward evolutionary multitasking. *IEEE Transactions on Evolutionary Computation* 20, 3 (2015), 343–357.

[22] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.

[23] Satoru Iwasaki, Heng Xiao, Toshiharu Hatanaka, and Takeshi Uchitane. 2017. A General Swarm Intelligence Model for Continuous Function Optimization. In *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 972–980.

[24] Chen Jin, Pei-Wei Tsai, and Alex Kai Qin. 2019. A Study on Knowledge Reuse Strategies in Multitasking Differential Evolution. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1564–1571.

[25] Nur Farahlina Johari, Azlan Mohd Zain, Noorfa Haszlinna Mustaffa, and Amirmudin Udin. 2017. Machining parameters optimization using hybrid firefly algorithm and particle swarm optimization. In *Journal of Physics: Conference Series*, Vol. 892. 012005.

[26] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient

global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4 (1998), 455–492.

[27] Tomas Kadavy, Michal Pluhacek, Adam Viktorin, and Roman Senkerik. 2018. Multi-swarm Optimization Algorithm Based on Firefly and Particle Swarm Optimization Techniques. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, 405–416.

[28] İH Karaçizmeli, İB Aydilek, A Gümüşçü, ME Tenekeci, and S Kaya. 2019. Evaluating Solution Performance of Hybrid Firefly and Particle Swarm Optimization Algorithm in Flow Shop Scheduling Problems. *Proceedings Book* (2019), 126.

[29] James Kennedy. 2006. Swarm intelligence. In *Handbook of nature-inspired and innovative computing*. Springer, 187–219.

[30] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, Vol. 4. IEEE, 1942–1948.

[31] Takehisa Kohira, Hiromasa Kemmotsu, Oyama Akira, and Tomoaki Tatsukawa. 2018. Proposal of benchmark problem based on real-world car structure design optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 183–184.

[32] Padmavathi Kora and K Sri Rama Krishna. 2016. Hybrid firefly and particle swarm optimization algorithm for the detection of bundle branch block. *International Journal of the Cardiovascular Academy* 2, 1 (2016), 44–48.

[33] Manasvi Kunapareddy and Bathina Venkateswara Rao. 2020. Hybridization of Particle Swarm Optimization with Firefly Algorithm for Multi-objective Optimal Reactive Power Dispatch. In *Innovative Product Design and Intelligent Manufacturing Systems*. Springer, 673–682.

[34] Er Deepam Sharma1 Dr Harish Kundra. 2016. Hybrid Algorithm Of Particle Swarm Optimization and Firefly for Natural Terrain Feature Extraction. *International Journal of Computer Science and Information Security (IJCSIS)* 14, 12 (2016).

[35] Akira Oyama, Takehisa Kohira, Hiromasa Kemmotsu, Tomoaki Tatsukawa, and Takeshi Watanabe. 2017. Simultaneous structure design optimization of multiple car models using the K computer. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1–4.

[36] Bijaya Ketan Panigrahi, Yuhui Shi, and Meng-Hiot Lim. 2011. *Handbook of swarm intelligence: concepts, principles and applications*. Vol. 8. Springer Science & Business Media.

[37] Rituraj Singh Patwal, Nitin Narang, and Harish Garg. 2018. A novel TVAC-PSO based mutation strategies algorithm for generation scheduling of pumped storage hydrothermal system incorporating solar units. *Energy* 142 (2018), 822–837.

[38] B Pitchaimanickam and G Murugaboopathi. 2019. A hybrid firefly algorithm with particle swarm optimization for energy efficient optimal cluster head selection in wireless sensor networks. *Neural Computing and Applications* (2019), 1–15.

[39] Jacob Robinson, Seelig Sinton, and Yahya Rahmat-Samii. 2002. Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In *IEEE Antennas and Propagation Society International Symposium (IEEE Cat. No. 02CH37313)*, Vol. 1. IEEE, 314–317.

[40] Mohamed AM Shaheen, SF Mekhamer, Hany M Hasanien, and Hossam EA Talaat. 2019. Optimal Power Flow of Power Systems Using Hybrid Firefly and Particle Swarm Optimization Technique. In *2019 21st International Middle East Power Systems Conference (MEPCON)*. IEEE, 232–237.

[41] Prakash Shelokar, Patrick Siarry, Valadi K Jayaraman, and Bhaskar D Kulkarni. 2007. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied mathematics and computation* 188, 1 (2007), 129–142.

[42] Yuhui Shi. 2011. Brain storm optimization algorithm. In *International conference in swarm intelligence*. Springer, 303–309.

[43] Serhiy D Shtovba. 2005. Ant algorithms: theory and applications. *Programming and Computer Software* 31, 4 (2005), 167–178.

[44] P Sivaranjani and A Senthil Kumar. 2017. Hybrid Particle Swarm Optimization-Firefly algorithm (HPSOFF) for combinatorial optimization of non-slicing VLSI floorplanning. *Journal of Intelligent & Fuzzy Systems* 32, 1 (2017), 661–669.

[45] Hui Song, Alex Kai Qin, Pei-Wei Tsai, and JJ Liang. 2019. Multitasking Multi-Swarm Optimization. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1937–1944.

[46] Zedong Tang and Maoguo Gong. 2019. Adaptive multifactorial particle swarm optimisation. *CAAI Transactions on Intelligence Technology* 4, 1 (2019), 37–46.

[47] Yuki Tanigaki, Naoki Masuyama, and Yusuke Nojima. 2018. Effects of the Number of Constraints on the Performance of Multi-Objective Evolutionary Algorithms. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND NETWORK SECURITY* 18, 12 (2018), 221–231.

[48] M Thirupathaiah. 2018. Enhancement of power quality in wind power distribution system by using hybrid PSO-firefly based DSTATCOM. *International Journal of Renewable Energy Research (IJRER)* 8, 2 (2018), 1138–1154.

[49] Guohua Wu, R Mallipeddi, and Ponnuthurai N Suganthan. 2017. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report* (2017).

[50] Xuewen Xia, Ling Gui, Guoliang He, Chengwang Xie, Bo Wei, Ying Xing, Ruifeng Wu, and Yichao Tang. 2018. A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm. *Journal of computational science* 26 (2018), 488–500.

[51] Heng Xiao and Toshiharu Hatanaka. 2016. Heterogenous swarm with first and second order dynamics for function optimization. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. 2077–2082.

[52] Heng Xiao and Toshiharu Hatanaka. 2016. Hybrid Swarm with Property Changing Particles for Function Optimization. In *Proceedings of the 7th International Symposium on Computational Intelligence and Industrial Applications*. 1–4.

[53] Heng Xiao and Toshiharu Hatanaka. 2021 (in press). Model selecting PSO-FA hybrid for Complex Function Optimization. *International Journal of Swarm Intelligence Research* 12, 4 (2021 (in press)).

[54] Heng Xiao, Masato Oi, and Toshiharu Hatanaka. 2016. Comparison between PSO and FA about Agents' Moving Direction. In *Proceedings of the 10th SICE Symposium on Computational Intelligence*. 35–38.

[55] Heng Xiao, Gen Yokoya, and Toshiharu Hatanaka. 2019. Multifactorial PSO-FA Hybrid Algorithm for Multiple Car Design Benchmark. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 1926–1931.

[56] Heng Xiao, Gen Yokoya, and Toshiharu Hatanaka. 2022 (in press). Multifactorial Particle Swarm Optimization Enhanced by Hybridization with Firefly Algorithm. *International Journal of Swarm Intelligence Research* 13, 1 (2022 (in press)).

[57] Tian Xie, Maoguo Gong, Zedong Tang, Yu Lei, Jia Liu, and Zhao Wang. 2016. Enhancing evolutionary multifactorial optimization based on particle swarm optimization. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1658–1665.

[58] Jianbin Xin, Guimin Chen, and Yubao Hai. 2009. A particle swarm optimizer with multi-stage linearly-decreasing inertia weight. In *2009 International Joint Conference on Computational Sciences and Optimization*, Vol. 1. IEEE, 505–508.

[59] Chunfan Xu and Haibin Duan. 2010. Artificial bee colony (ABC) optimized edge potential function (EPF) approach to target recognition for low-altitude aircraft. *Pattern Recognition Letters* 31, 13 (2010), 1759–1772.

[60] Guangyou Yang, Dingfang Chen, and Guozhu Zhou. 2006. A new hybrid algorithm of particle swarm optimization. In *International Conference on Intelligent Computing*. Springer, 50–60.

[61] Xin-She Yang. 2009. Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*. Springer, 169–178.

[62] Xin-She Yang. 2010. *Nature-inspired metaheuristic algorithms*. Luniver press.

[63] Xin-She Yang and Xingshi He. 2013. Bat algorithm: literature review and applications. *International Journal of Bio-inspired computation* 5, 3 (2013), 141–149.

[64] Mauricio Zambrano-Bigiarini, Maurice Clerc, and Rodrigo Rojas. 2013. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso

improvements. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2337–2344.

[65] Wen-Jun Zhang and Xiao-Feng Xie. 2003. DEPSO: hybrid particle swarm with differential evolution operator. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, Vol. 4. IEEE, 3816–3821.

[66] Xiaolong Zheng, Yu Lei, Maoguo Gong, and Zedong Tang. 2016. Multifactorial brain storm optimization algorithm. In *International Conference on Bio-Inspired Computing: Theories and Applications*. Springer, 47–53.

[67] Xiaolong Zheng, Alex Kai Qin, Maoguo Gong, and Deyun Zhou. 2019. Self-regulated evolutionary multi-task optimization. *IEEE Transactions on Evolutionary Computation* (2019).