applied sciences

MDPI

Article

Cyber Firefly Algorithm Based on Adaptive Memory Programming for Global Optimization

Peng-Yeng Yin [1,2,*], Po-Yen Chen [1], Ying-Chieh Wei [2] and Rong-Fuh Day [1]

[1] Department of Information Management, National Chi Nan University, No. 1, University Rd., Puli, Nantou 54561, Taiwan; s100213516@ncnu.edu.tw (P.-Y.C.); rfday@ncnu.edu.tw (R.-F.D.)
[2] Institute of Strategy and Development of Emerging Industry, National Chi Nan University, No. 1, University Rd., Puli, Nantou 54561, Taiwan; s103245908@ncnu.edu.tw
* Correspondence: pyyin@ncnu.edu.tw

Received: 12 October 2020; Accepted: 14 December 2020; Published: 15 December 2020

Abstract: Recently, two evolutionary algorithms (EAs), the glowworm swarm optimization (GSO) and the firefly algorithm (FA), have been proposed. The two algorithms were inspired by the bioluminescence process that enables the light-mediated swarming behavior for mating or foraging. From our literature survey, we are convinced with much evidence that the EAs can be more effective if appropriate responsive strategies contained in the adaptive memory programming (AMP) domain are considered in the execution. This paper contemplates this line and proposes the Cyber Firefly Algorithm (CFA), which integrates key elements of the GSO and the FA and further proliferates the advantages by featuring the AMP-responsive strategies including multiple guiding solutions, pattern search, multi-start search, swarm rebuilding, and the objective landscape analysis. The robustness of the CFA has been compared against the GSO, FA, and several state-of-the-art metaheuristic methods. The experimental result based on intensive statistical analyses showed that the CFA performs better than the other algorithms for global optimization of benchmark functions.

Keywords: adaptive memory programming; firefly algorithm; global optimization; glowworm swarm optimization; metaheuristics

1. Introduction

Many challenging problems in modern engineering and business domains challenge the design of satisfactory algorithms. Traditionally, researchers resort to either mathematical programming approaches or heuristic algorithms. However, mathematical programming approaches are plagued by the curse of problem size and the heuristic algorithms have no guarantees to near-optimal solutions. Recently, metaheuristic approaches have come as an alternative between the two extreme approaches. The metaheuristic approaches can be classified into two classes, evolution-based and memory-based algorithms. The evolutionary algorithms (EAs) iteratively improve solution quality by decent operations inspired by nature metaphors, creating several novel algorithms, such as genetic algorithms, artificial immune systems, ant colony optimization, and particle swarm optimization. The memory-based metaheuristic approaches guide the search course to go beyond the local optimality by taking full advantage of adaptive memory manipulations. Typical renowned methods in this class include tabu search, path relinking, scatter search, variable neighborhood search, and greedy randomized adaptive search procedures (GRASP).

The metaheuristic approaches contained in the two classes have developed rather independently, and only a few works investigate the possible synergy between them [1]. Talbi and Bachelet [2] proposes the COSEARCH approach which combines the tabu search and a genetic algorithm for solving the quadratic assignment problem. Shen et al. [3] proposes an approach called HPSOTS

which enables the particle swarm optimization to circumvent local optima by restraining the particle movement based on the use of tabu memory. A hybrid of the ant colony optimization and the GRASP is proposed by Marinakis et al. [4] for cluster analysis. However, the two metaheuristics are separately used to tackle the feature selection and clustering problems, respectively. Fuksz and Pop [5] proposes a hybrid genetic algorithm (GA) with variable neighborhood search (VNS) to the number partitioning problem. Their hybrid GA-VNS runs the GA as the main algorithm and the VNS procedure for improving individuals within the population. Yin et al. [6] introduces the cyber swarm algorithm which gives more substance to the particle swarm optimization (PSO) by incorporating the adaptive memory programming (AMP) strategies introduced in the scatter search and path-relinking (SS/PR) template. The adjective "cyber" emphasizes the connection between the evolutionary swarm metaheuristics and the AMP metaheuristics. The cyber swarm algorithm outperforms several state-of-the-art metaheuristics on complex benchmark functions. The experimental results of the above-noted works disclose a promising research area that the marrying of the approaches from the two classes of metaheuristics can create significant benefit that is hardly obtained by the approaches from each class alone.

More recently, two evolution-based algorithms [7,8], namely the glowworm swarm optimization (GSO) and the firefly algorithm (FA), were proposed. The two algorithms were inspired by the bioluminescence process that enables the light-mediated swarming behavior for mating or foraging. The intensity of the light and the distance between the light source and the observer determine the attractiveness degree that causes the moving maneuver. This form of metaphor can be used to develop a swarm-based optimization algorithm where a swarm of glowworms/fireflies represent a set of candidate solutions. The light intensity is evaluated by the objective value of the light source and the distance between the glowworms/fireflies implicitly defines the eligible neighbors since the light observed by an agent decays with the distance. The glowworms/fireflies are attracted by visible light sources and fly towards them. Therefore, the solutions represented by the glowworms/fireflies improve their objective value through the swarming behavior. Several improvements of FA have been proposed. Yang proposed the LFA [9] by combining his original FA with Levy flight. Yu et al. employed the variable step size strategy to create the VESSFA [10] variant. The dynamic adaptive inertia weight was adopted in WFA [11] to use the short-term memory of previous moving velocity. Kaveh et al. developed CLFA [12] which applies chaos theory and logical mapping to determine the optimal FA parameters. The Tidal Force formula was used in FAtidal [13] to improve the balance between exploitation and exploration search behaviors. The most recent improvement was GDAFA [14] which uses global-oriented positional update and dynamically adjusts the step size and attractiveness to avoid being trapped in local optima. The DsFA [15] employed dynamic step change strategy to balance the global and local search capabilities, such that the search course is not likely trapped in local optimum.

From a long-term perspective of metaheuristic development, we anticipate that the GSO and the FA can be made more effective by incorporating the notions from the memory-based approaches, as validated in many previous attempts. Based on the prevailing framework of the cyber swarm algorithm [6] that integrates key elements of the two types of metaheuristic methods, we propose the Cyber Firefly Algorithm (CFA) to proliferate the advantages of the original form of the GSO and the FA. The CFA conceptions include the employment of multiple guiding fireflies, the embedding of the pattern search, firefly swarm rebuilding by the multi-start search, and the responsive strategies based on objective landscape analysis. The robustness of the CFA has been compared against the GSO, FA, and several state-of-the-art metaheuristic methods. The result as demonstrated in our statistical analyses and comprehensive experiments showed the CFA manifests a more effective form of GSO and FA. Most FA variants intend to improve the position update mechanism such as by using Levy flight [9], variable step size strategy [10], adaptive inertia weight [11], and dynamically adjusting strategy [14]. Our CFA differs with these variants by facilitating a more intelligent step size control mechanism which performs landscape analysis in the objective space to adaptively tune the step size according to the profiles of the incumbent fitness landscape.

The novelty of this paper stems at creating a more effective form of GSO and FA approaches by bridging the advantages of evolution-based and memory-based metaheuristics. In particular, this paper investigates whether the CSA template is viable for improving GSO and FA as CSA has been shown in [6] for improving PSO. Our experimental results show that the proposed CFA prevails GSO, FA, and several state-of-the-art metaheuristics on benchmark datasets. This justifies the generalization capability of the CSA template and one can follow the template to create an effective version of interested metaheuristics.

The remainder of this paper is organized as follows. Section 2 presents a literature review and Section 3 proposes the CFA and the employed features. Section 4 presents the results of intensive experiments. Finally, conclusions and future research possibilities are given in Section 5.

2. Related Work Materials and Methods

2.1. Glowworm Swarm Optimization (GSO)

Krishnanand and Ghose [7] developed the GSO algorithm. It is assumed that each glowworm has a luciferin level and a local visibility range. The luciferin level of a glowworm determines its light intensity, and the local visibility range identifies the neighboring glowworms which are visible to it. The glowworm probabilistically chooses a neighbor which has a higher luciferin level than itself and is flying towards this neighbor due to light attraction. The local visibility range is dynamic for maintaining an ideal number of neighbors. The GSO simulates the glowworm behaviors and consists of three phases as depicted as follows.

The luciferin update phase evaluates the luciferin level of every glowworm according to the decay of its luminescence and the merit of its new position after performing the movement within the evolution cycle t. The luciferin level of glowworm i is updated by the following equation.

$$l_i^{t+1} \leftarrow (1-\rho) l_i^t + \tau f_i^{t+1} \tag{1}$$

where ρ is the decay ratio of the glowworm's luminescence and τ is an enhancement constant. The first term is the persistence substance of luminescence due to decay with time, and the second term is the additive luminescence as a function of f_i^{t+1} which indicates the objective value measured at the glowworm's new position (here, without loss of generality, we assume the objective function is to be maximized).

In the movement phase of each evolution cycle, every glowworm in the swarm must perform a movement by flying towards a neighbor which has a higher luciferin level than the incumbent glowworm and is located within the local visibility neighborhood defined by a radius r_i^t. The probability with which glowworm i is attracted to a brighter glowworm j at evolution cycle t is given by:

$$p_{ij} = \frac{l_j^t - l_i^t}{\sum_{k \in N_i^t} l_k^t - l_i^t} \tag{2}$$

where N_i^t is the set of glowworms within the visibility neighborhood of glowworm i at evolution cycle t. Once selecting a neighbor, say glowworm j, the current glowworm i performs a movement to update its position as follows.

$$x_i^{t+1} \leftarrow x_i^t + s\left(\frac{x_j^t - x_i^t}{\|x_j^t - x_i^t\|}\right) \tag{3}$$

where s is the movement distance and $\|\bullet\|$ indicates the length of the referred vector. Precisely speaking, glowworm i moves in s units of distance towards glowworm j.

The visibility range update phase dynamically tunes the visibility radius of each glowworm to maintain an ideal number of neighbors, N^*. So, the current number of neighbors, $\left|N_i^t\right|$, is compared to N^* and the visibility radius r_i^t is tuned by the following equation.

$$r_i^{t+1} \leftarrow \min\left\{r_{\max},\ \max\left\{0,\ r_i^t + \eta\left(N^* - N_i^t\right)\right\}\right\} \tag{4}$$

where $r_{\max}$ is the maximum visibility radius and η is a scaling parameter for tuning N_i^t. Therefore, the value of r_i^t is increased if $N_i^t < N^*$, and it is decreased if $N_i^t > N^*$. The feasible range of r_i^t is bounded between $[0, r_{\max}]$. The phenomenon of $r_i^t = 0$ indicates many glowworms are resorting to the position of the current glowworm, while $r_i^t = r_{max}$ discloses the situation that the current glowworm is in a large distance to most of the other glowworms.

2.2. Firefly Algorithm (FA)

Yang [8] introduced the FA. The FA explores the solution space with a population of fireflies. Each firefly has luminescence of flashing light which attracts its mates in an inverse multiplication of the squared distance and the light absorption. By using the metaphor, a firefly (representing a candidate solution) can improve its light intensity (a merit function of the objective value) by flying toward a more attractive firefly. In particular, the attractiveness of firefly j observed by firefly i is defined as follows.

$$\beta_r = \beta_0 e^{-\gamma r^2} \tag{5}$$

where γ is the light absorption coefficient, r is the Euclidean distance between the two fireflies, and β_0 is the light intensity of firefly j.

The movement of firefly i is attracted to a more attractive firefly j by the following equation,

$$x_i^{t+1} \leftarrow x_i^t + \beta_r\left(x_j^t - x_i^t\right) + \alpha\varepsilon_i \tag{6}$$

where the second term is due to the light attraction and the third term is a random perturbation with α being the randomization parameter and ε_i is a vector of small random numbers drawn from a Gaussian distribution or uniform distribution. After the movement, the light intensity of firefly i should be re-evaluated and the relative attractiveness of any other flies to firefly i is also re-calculated. The FA conducts a maximum number of evolution cycles and within each cycle every pair of fireflies should be examined for possible movement due to attraction and randomization.

2.3. Adaptive Memory Programming (AMP)

The AMP comprises a broader spectrum than its more popularly accepted branch, the tabu search [16]. In what follows, we focus our discussion on the use of the SS/PR template [17] which we found very effective in creating benefits for marrying with evolutionary-based metaheuristics.

2.3.1. Scatter Search (SS)

The scatter search (SS) [18] operates on a common reference set consisting of diverse and elite solutions observed throughout the evolution history. The SS method dynamically updates the reference set and systematically selects subsets of the reference set to generate new solutions. These new solutions are improved until local optima are obtained and the reference set is updated by comparing its current members to these local optima. Some important features of the SS are as follows. (1) A diversification-generation method is designed to identify the under-explored region in the solution space such that a set of diverse trial solutions can be produced. (2) An improvement method is used to enhance the quality of the solutions under keeping. This process usually involves a local search operation which brings the trial solution to a local optimum. (3) A reference set update method is adopted to make sure that the reference set is maintaining a set of high quality and mutually diverse solutions observed overall in search history. (4) The multi-start search strategy iteratively

restarts a new search session when the current search loses it efficacy. To lead the search course towards under-explored solution space, the multi-start strategy works with a rebuilt reference set produced by the diversification-generation method. (5) Interactions between multiple reference solutions are systematically contemplated. The simplest implementation is to generate all 2-element subsets consisting of exactly two reference solutions. Various search courses are conducted between and beyond the selected reference solutions.

2.3.2. Path Relinking (PR)

There is a common hypothesis accepted by most of the metaheuristics that elite solutions often lie on trajectories from good solutions to other good solutions. In a broader sense, the crossover of chromosomes, the sociocognition learning of particles, and the pheromone trail searching are all effective operators following the noted hypothesis. PR therefore creates a search path between elite solutions. An initiating solution and a guiding solution are selected from the repository of elite solutions. PR then transforms the initiating solution into the guiding solution by generating a succession of moves that introduce attributes from the guiding solution into the initiating solution. The relinked path can go beyond the guiding solution to extend the search trajectory. PR works in the neighborhood space instead of the Euclidean space and variable neighborhoods are usually considered in performing successive moves. Therefore, PR is well fitted for use as a diversification strategy.

3. Proposed Methods

Our proposed CFA synergizes the strength from three domains, namely the GSO, the FA, and the AMP, to create a more effective global optimization metaheuristic algorithm. We articulate the new features of the CFA as follows.

3.1. Multiple Guiding Points

Both GSO and FA algorithms conduct the firefly movement by referring to a guiding point, which is a nearby elite firefly with a higher fitness than that of the moving firefly. This mechanism of using a single guiding point raises the risk of misleading due to the selection of a false peak. Our proposed CFA algorithm instead employs a two-guiding-point mechanism to enhance the exploration capability of the algorithm. More precisely, the neighboring fireflies of the incumbent firefly i are identified by reference to the current value of the visibility radius, r_i^t, where t indicates the index of the evolution iteration. Among the neighbors, the elite fireflies with a higher fitness than that of the incumbent firefly are eligible for guiding-point selection. We employ the rank selection strategy to select two guiding points from the eligible neighbors. According to our preliminary experiments, the rank selection strategy is more robust against prickly fitness landscape than the roulette-wheel selection strategy because the former works in the ranking value space instead of the fitness space.

3.2. Luciferin-Proportional Movement

In GSO and FA, the luciferin of firefly is used only for the determination of the single guiding point. As we deploy the two-guiding-point mechanism, the relative significance of contribution from each of the two guiding points can be further elaborated. We propose to convert the luciferin of each guiding point to the weighting value for individual contribution. Given two guiding points x_j^t and x_k^t observed at iteration t, their weighting values ω_1 and ω_2 are set as follows.

$$\omega_1 = \frac{l_j^t}{l_j^t + l_k^t} \tag{7}$$

$$\omega_2 = \frac{l_k^t}{l_j^t + l_k^t} = 1 - \omega_1 \tag{8}$$

We then propose the firefly movement formula as follows.

$$x_i^{t+1} \leftarrow x_i^t + \varphi_1\omega_1\beta_{r_j}\left(x_j^t - x_i^t\right) + \varphi_2\omega_2\beta_{r_k}\left(x_k^t - x_i^t\right) \quad (9)$$

where φ_1 and φ_2 are random values drawn from a uniform distribution U(lb, ub). Hence, firefly i is drawn by firefly j and firefly k with driving force relative to respective luciferin level. Parameters lb and ub control the feasible range of the step size for every firefly movement and their values are dynamically tuned in accordance with the landscape analysis as will be noted.

3.3. Adaptive Local Search

Local search is a rudimentary component contained in most of the successful metaheuristics such as memetic algorithms [19], scatter search [18], and GRASP [20], to name a few. Our CFA employs the pattern search method [21,22] which iteratively performs a two-step trajectory search. The explorative search step tentatively looks for an improving neighbor while the pattern search step aggressively expands the improving move by doubling the execution of the move. The proposed CFA embodies the pattern search method as a local search but performs it in an adaptive way. The local search is performed on all fireflies only after consuming every t_1 fitness function evaluations. To leverage the balance between search efficiency and effectiveness, we adaptively vary the frequency parameter (t_1) for the performed local search. The adaptive local search strategy is based on the analysis of the landscape observed in the objective space as will be noted.

3.4. Multi-Start with PR for Firefly Swarm Rebuilding

Multi-start is a mechanism for reinitiating the search with an under-explored region when the trajectory search loses its efficacy. The multi-start search strategy thus works with two functions. The function for *critical event detection* emits a signal upon the moment when the search stagnation behavior is observed. The *diversification-generation* function identifies an under-explored region in the solution space to generate a starting solution for the new trajectory search. Our CFA approach facilitates the critical event detection by monitoring the number of enduring iterations since the last improvement of the best-so-far firefly. If the number of no-improvement iterations for the best-so-far firefly has exceeded a parameter t_2, a signal for detection of a critical event is returned. The parameter t_2 is also made adaptive by the landscape analysis approach.

The diversification-generation function of the CFA approach is implemented by using the path-relinking technique, which has been found very useful in identifying a promising solution within an under-explored region and has been adopted as a diversification strategy such as in Yin et al. [6]. When a critical event signal is activated, a ratio δ of the swarm fireflies are rebuilt and each new firefly is placed on the best solution along a relinked path connecting a random solution and the best-so-far firefly. The path-relinking technique creates the path by dividing the subspace between the two end points of the path into n equal-size hyper-grids where n is the number of decision variables for the addressed optimization problem. A solution is sampled within each hyper-grid, so in total n solutions will be obtained along the relinked path. The best of the n sampled solutions is designated as the rebuilt firefly.

3.5. Responsive Strategies Based on Landscape Analysis

The previous features of CFA can be more effective by incorporating the notion of responsive strategy which adaptively tunes the search strategy when observing the status transitions. The CFA adapts the strategy parameters when the search encounters the transition between the smooth and prickly landscapes manifested by the objective function. The measure of fitness distance correlation (FDC) proposed by Jones and Forrest [23] has been proved useful for judging the suitability of a fitness landscape for various search algorithms. The FDC measures the correlation between the solution cost

and the distance to the closest global optimum. Let the set of observed cost-distance pairs be $\{(c_1, d_1), \ldots, (c_m, d_m)\}$, the correlation coefficient is defined as

$$\mathrm{FDC} = \frac{\frac{1}{m}\sum_{i=1}^{m}\left(c_i - \bar{c}\right)\left(d_i - \bar{d}\right)}{\sigma_c \sigma_d} \tag{10}$$

where $\bar{c}$ and $\bar{d}$ are the mean cost and the mean distance, and σ_c and σ_d. are the standard deviations of the costs and distances. Hence, a significant FDC value has the implication that on average, the better the solution quality the closer this solution is to an optimal solution. It can be contemplated that a single-modal objective function would impose a high FDC value in contrast to a multi-modal objective function whose FDC value is closer to zero because of the irrelevance between the fitness and the distance.

Most existing research adopted the FDC technique as an off-line analysis for realizing the characteristics of the fitness landscape, while our CFA exploits the fitness landscape analysis as an online responsive strategy. We periodically conduct the fitness landscape analysis for every $1000n$ fitness function evaluations, where n is the number of problem variables. All the visited solutions ever identified as local optima in the trajectory of a firefly within this time period are eligible for the FDC value computation because these local optima are representative solutions produced in the evolution. As the global optimum is unknown, the best solution obtained at the end of this period is considered to be the best-known solution. The FDC value is computed by using these local optima and the best solution identified in the current time period of previous $1000n$ fitness function evaluations. Consequently, our CFA performs the adaptive landscape analysis which can predict the objective landscape within the current region explored by the firefly swarm and activate appropriate responsive strategies to make the search more effective. If the derived FDC value for the current time period is greater than a threshold (FDC > h_1), the local objective landscape is considered to be asymptotical single-modal, and we perform two responsive strategies as follows: (1) increasing the distance, on average, of the firefly movement by $lb = lb/\lambda$ and $ub = ub/\lambda$ ($0 < \lambda < 1$); (2) increasing the elapsed iterations for checking the feasibility of executing the local search and multi-start search by $t_1 = t_1/\lambda$ and $t_2 = t_2/\lambda$. The implication of the two responsive strategies is that when the local objective landscape is single-modal, the search may be more effective if the firefly movement distance is greater and the frequency for conducting the local search and the multi-start search is lower. On contrary, If the absolute FDC value for the current time period is less than a threshold (|FDC| < h_2), the local objective landscape is considered to be asymptotical multi-modal, and we perform two responsive strategies as follows: (1) decreasing the distance, on average, of the firefly movement by $lb = lb \times \lambda$ and $ub = ub \times \lambda$; (2) decreasing the elapsed iterations for checking the feasibility of executing the local search and multi-start search by $t_1 = t_1 \times \lambda$ and $t_2 = t_2 \times \lambda$.

3.6. Pseudo-Code of CFA

The pseudo-code of our CFA is elaborated in Figure 1. The new features of the CFA are emphasized in boldface for comprehensive descriptions.

1 Initialize.
  1.1 Set the initial value of the algorithmic parameters.
  1.2 Generate the initial swarm of Ns fireflies at random.
  1.3 Evaluate the fitness of each firefly.
2 Repeat until Max_FE fitness function evaluations have been executed.
  2.1 For each firefly x_i^t, perform **rank selection** to select two local guides x_j^t and x_k^t from those fireflies which are brighter than x_i^t and within its visibility range.
    2.1.1 perform the **luciferin-proportional** and **multi-guider** firefly movement by
$$x_i^{t+1} \leftarrow x_i^t + \varphi_1\omega_1\beta_{r_j}\left(x_j^t - x_i^t\right) + \varphi_2\omega_2\beta_{r_k}\left(x_k^t - x_i^t\right)$$
    2.1.2 Evaluate the fitness of firefly x_i^{t+1}
    2.1.3 After every period of t_1 consecutive iterations, perform the **adaptive local search** on x_i^{t+1} until a local optimum is reached.
    2.1.4 Update the visibility radius by
$$r_i^{t+1} \leftarrow \min\left\{r_{\max}, \max\left\{0, r_i^t + \eta\left(N^* - N_i^t\right)\right\}\right\}$$
  2.2 **Multi-Start**: If the best-so-far firefly has not improved for t_2 consecutive iterations, rebuild a ratio δ of the firefly swarm. Each rebuilt firefly is produced by
$$x_i^{t+1} \leftarrow path\_relinking\left(random\ firefly, best - so - far\ firefly\right)$$
  2.3 **Responsive Strategies**: After every period of $1000n$ iterations, conduct the fitness landscape analysis and adaptively tune the parameters, including average moving distance, local search frequency, and multi-start frequency.

Figure 1. Pseudo-code of the CFA algorithm.

4. Experimental Results and Discussions

We have conducted intensive experiments and statistical tests to compare the performance of the proposed CFA algorithm and its counterparts. The experimental results disclose several interesting outcomes in addition to establishing the effectiveness of the proposed method. The platform for conducting the experiments is a PC with an Intel Core i5CPU and 8.0 GB RAM. All programs are coded in C# language.

4.1. Benchmark Test Functions and Algorithm Parameter Settings

We have chosen two benchmark datasets. (1) The standard benchmark dataset contains 23 test functions that are widely used in the nonlinear global optimization literature [6,18,24,25]. The detailed function formulas can be found in the relevant literature and they have a wide variety of different landscapes and present a significant challenge to optimization methods. The number of variables, domain, and optimal value of these benchmark test functions are listed in Table 1. Performance evaluation on this dataset is reported in terms of the mean best function value obtained from 30 repetitive runs. For each run, the tested algorithm can perform 160,000 function evaluations (FE) for ensuring that the tested algorithm has very likely converged. (2) The IEEE CEC 2005 benchmark dataset which is designed for unconstrained real-parameter optimization. We selected the most challenging functions and compared our CFA with the best methods reported in [26] under the same evaluation criteria described in the original paper. The implementation of the benchmark functions from both datasets is available in public library [27] in some programming languages such as MATLAB and Java. As we used C# for all the experiments, we modified the public codes to C# and embedded them into our main program.

Table 1. The number of variables, domain, and optimal value of the benchmark functions.

No.	Function Name	Number of Variables (*r*)	Domain	Optimal Value
1	Easom (2)	2	[−10, 10]	−1.0
2	Shubert (2)	2	[−10, 10]	−186.7309
3	Rosenbrock (2)	2	[−30, 30]	0.0
4	Zakharov (2)	2	[−5, 10]	0.0
5	De Jong (3)	3	[−5.12, 5.12]	0.0
6	Shekel (4, 5)	4	[0, 10]	−10.1532
7	Shekel (4, 7)	4	[0, 10]	−10.4029
8	Shekel (4, 10)	4	[0, 10]	−10.5364
9	Sphere (10)	10	[−100, 100]	0.0
10	Rosenbrock (10)	10	[−30, 30]	0.0
11	Rastrigin (10)	10	[−5.12, 5.12]	0.0
12	Griewank (10)	10	[−600, 600]	0.0
13	Zakharov (10)	10	[−5, 10]	0.0
14	Sphere (20)	20	[−100, 100]	0.0
15	Rosenbrock (20)	20	[−30, 30]	0.0
16	Rastrigin (20)	20	[−5.12, 5.12]	0.0
17	Griewank (20)	20	[−600, 600]	0.0
18	Zakharov (20)	20	[−5, 10]	0.0
19	Sphere (30)	30	[−100, 100]	0.0
20	Rosenbrock (30)	30	[−30, 30]	0.0
21	Rastrigin (30)	30	[−5.12, 5.12]	0.0
22	Griewank (30)	30	[−600, 600]	0.0
23	Zakharov (30)	30	[−5, 10]	0.0

To obtain the parameter settings of the CFA, GSO, and FA, various values for their parameters were tested with the standard benchmark dataset and the values that resulted in the best mean function value were used as the parameter settings as tabulated in Table 2. The parameter settings used by other compared metaheuristics will be presented in Sections 4.3.2 and 4.3.3 where the comparative experiments are presented.

Table 2. Parameter settings of the CFA, GSO, and FA.

Notation	Parameter Description	Set Value
	Common parameters of CFA, GSO, and FA:	
Num_Run	Number of independent runs	30
Max_FE	Maximum number of executed FE for each run	160,000
N_S	Number of fireflies	60
N^*	Ideal number of local neighbors	10
L	Lower bound of decision variable	See Domain in Table 1
U	Upper bound of decision variable	See Domain in Table 1
r_{max}	Maximum visibility radius of fireflies	$(U - L) \times 0.05$
	Additional parameters used by CFA:	
lb	Initial minimum step size (to be adaptively tuned)	$(U - L) \times 10^{-6}$
ub	Initial maximum step size (to be adaptively tuned)	$(U - L) \times 10^{-2}$
t_1	Number of evolution iterations for activating adaptive local search (to be adaptively tuned)	20
t_2	Number of non-improving iterations for activating multi-start search (to be adaptively tuned)	50
δ	Ratio of swarm fireflies to be rebuilt	0.3
h_1	FDC single-modal threshold	0.5
h_2	FDC multi-modal threshold	0.4
λ	Scaling factor for performing responsive strategies	0.5

4.2. Analysis of CFA Strategies

To understand the influence on performance of using various CFA strategies, we conduct experiments on the benchmark functions with several CFA variants as in the following subsections.

4.2.1. Selection Strategy for Multiple Guiding Solutions

In contrast to GSO and FA, the CFA selects multiple guiding solutions for performing the firefly movement. The firefly can circumvent the false peaks by relaxing the limitation that constrains the firefly to move toward the single best solution within neighborhood. To investigate the influence on performance of using various selection strategies for guiding solutions, we compare the variants of the CFA that employs the roulette-wheel selection, the tournament selection, and the rank selection, respectively. The comparative performance of the CFA variants is listed in Table 3. It is seen that for the test functions with fewer than 10 variables and the relatively simple functions (Sphere and Zakharov), all of the CFA variants work well and the mean best value obtained is very near the optimal value. For the harder and larger functions (Rosenbrock, Rastrigin, and Griewank with 10 or more variables), the CFA with the rank selection strategy finds a more effective solution for most of the functions than the CFA with the other two selection strategies. The rank selection outperforms the other selection strategies in tackling harder problems because it eliminates the fitness-scaling problem by working in the rank-value space instead of in the function-value space such that the firefly will not be severely misled by false but higher peaks.

Table 3. The mean best function value and the standard deviation obtained by using various selection strategies.

Functions	Roulette-Wheel Selection		Tournament Selection		Rank Selection	
	Mean	Std	Mean	Std	Mean	Std
Easom (2)	−1.0	0.0	−1.0	0.0	−1.0	0.0
Shubert (2)	−186.726	0.011381	−186.723	0.010061	−186.723	0.016331
Rosenbrock (2)	0.0	0.0	0.0	0.0	0.0	0.0
Zakharov (2)	0.0	0.0	0.0	0.0	0.0	0.0
De Jong (3)	0.0	0.0	0.0	0.0	0.0	0.0
Shekel (4, 5)	−8.74128	2.436384	−8.64565	2.5462	−8.09613	3.003044
Shekel (4, 7)	−9.91291	1.345608	−10.1835	1.11003	−10.2611	0.440954
Shekel (4, 10)	−10.532	0.013263	−10.511	0.130888	−10.3554	0.966638
Sphere (10)	0.0	0.0	0.0	0.0	0.0	0.0
Rosenbrock (10)	0.000052	0.000031	0.000206	0.000982	0.002511	0.009392
Rastrigin (10)	0.851670	0.878212	0.847894	0.930433	0.646338	0.800856
Griewank (10)	0.023954	0.015559	0.018670	0.010681	0.016450	0.009914
Zakharov (10)	0.0	0.0	0.0	0.0	0.0	0.0
Sphere (20)	0.0	0.0	0.0	0.0	0.0	0.0
Rosenbrock (20)	0.800503	1.594902	0.953237	1.679062	0.267050	0.994099
Rastrigin (20)	3.052760	1.833682	2.607486	1.544556	2.466102	1.765681
Griewank (20)	0.013792	0.020267	0.011079	0.010911	0.007213	0.012479
Zakharov (20)	0.0	0.0	0.0	0.0	0.0	0.0
Sphere (30)	0.0	0.0	0.0	0.0	0.0	0.0
Rosenbrock (30)	5.924423	3.057484	2.962683	2.773249	0.801094	1.598004
Rastrigin (30)	4.204611	2.066075	4.973816	1.826780	4.680690	2.632701
Griewank (30)	0.009349	0.012921	0.008202	0.012808	0.003364	0.007581
Zakharov (30)	0.000005	0.000005	0.0	0.0	0.0	0.0

4.2.2. Adaptive Strategy for Local Search

Local search is a rudimentary component contained in most of the modern metaheuristic algorithms. The local search procedure exploits the regional function profiles and makes the master metaheuristic algorithm more effective than the original form which does not embed this procedure. However, the local search procedure can be computationally expensive if it is performed within each iteration of

the master metaheuristic algorithm. As previously noted, the CFA applies the adaptive local search strategy which dynamically varies the frequency of the local search execution according to the result of the landscape analysis. Table 4 tabulates the mean best function value and the standard deviation obtained by CFA with or without adaptive local search. We observe that for all the benchmark test functions the CFA with adaptive local search obtains better or equivalent mean function value than its counterpart without adaptive local search.

Table 4. The mean best function value and the standard deviation obtained by CFA with or without adaptive local search.

Functions	No Adaptive Local Search		With Adaptive Local Search	
	Mean	Std	Mean	Std
Easom (2)	−0.99997	0.000020	−1.0	0.0
Shubert (2)	−185.779	0.963969	−186.723	0.016331
Rosenbrock (2)	0.0	0.0	0.0	0.0
Zakharov (2)	0.0	0.0	0.0	0.0
De Jong (3)	0.0	0.0	0.0	0.0
Shekel (4, 5)	−6.50369	2.613879	−8.09613	3.003044
Shekel (4, 7)	−7.40191	1.802712	−10.2611	0.440954
Shekel (4, 10)	−7.80353	1.033856	−10.3554	0.966638
Sphere (10)	0.0	0.0	0.0	0.0
Rosenbrock (10)	0.797320	1.594630	0.002511	0.009392
Rastrigin (10)	0.734548	0.827677	0.646338	0.800856
Griewank (10)	0.015448	0.011793	0.016450	0.009914
Zakharov (10)	0.0	0.0	0.0	0.0
Sphere (20)	0.0	0.0	0.0	0.0
Rosenbrock (20)	0.797828	1.594400	0.267050	0.994099
Rastrigin (20)	2.590780	1.391706	2.466102	1.765681
Griewank (20)	0.012054	0.013121	0.007213	0.012479
Zakharov (20)	0.0	0.0	0.0	0.0
Sphere (30)	0.0	0.0	0.0	0.0
Rosenbrock (30)	0.801375	1.599876	0.801094	1.598004
Rastrigin (30)	4.704743	1.842029	4.680690	2.632701
Griewank (30)	0.005663	0.007891	0.003364	0.007581
Zakharov (30)	0.0	0.0	0.0	0.0

4.2.3. Multi-Start Strategy for Swarm Rebuilding

Multi-start is a diversification strategy which terminates an ineffective trajectory search and reinitiates a new one from an under-explored region. Our CFA applies the adaptive multi-start strategy which monitors the performance of the incumbent firefly swarm. If the program stops improving the performance for a threshold number of consecutive iterations, part of the swarm is rebuilt by positioning some fireflies on diversified locations using the path-relinking technique. The performance stagnation threshold is made adaptive according to the result of the landscape analysis such that the frequency of the multi-start activation depends on the regional function profiles under exploration. It can be seen from Table 5 that the multi-start strategy effectively improves the mean best function value and the standard deviation obtained by CFA.

4.2.4. Responsive Strategies Based on Landscape Analysis

As noted in Section 3.5, the responsive strategies employed by the CFA are made adaptive based on the analysis of function landscape. The landscape analysis makes distinctions of two classic function forms: single-modal and multi-modal. The responsive strategies then vary the movement step size and the frequency of the local search and the multi-start activations to make the CFA search more effective. Table 6 lists the comparative performance of the CFA with or without performing the landscape analysis. It is noted that the version of the CFA without adaptive landscape analysis still embeds the local

search and the multi-start procedures as its rudimentary components, which, however, are executed with fixed parameter values. From the tabulated result, we conclude that the landscape analysis can proliferate the performance gains possibly obtained by the local search and the multi-start strategies.

Table 5. The mean best function value and the standard deviation obtained by CFA with or without multi-start strategy.

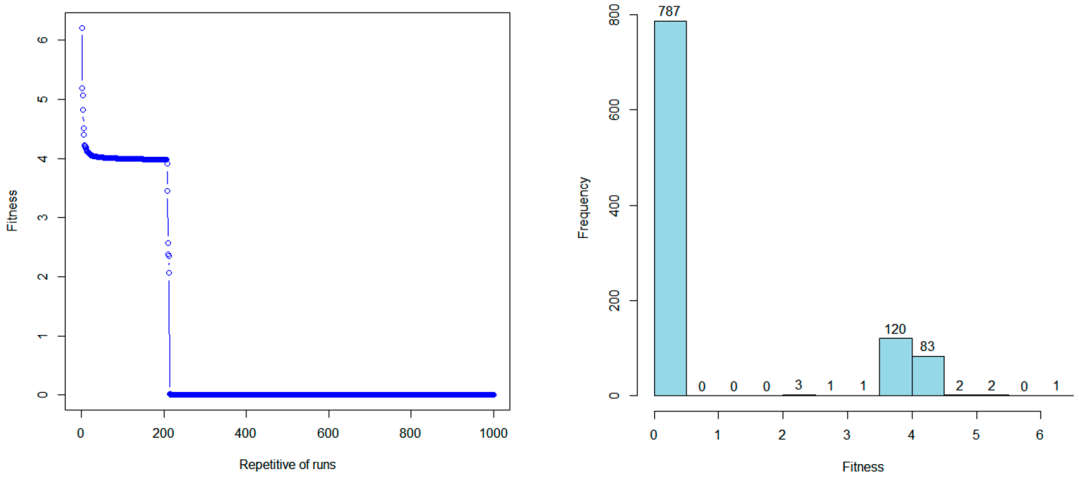
Functions	No Multi-Start Rebuilding		With Multi-Start Rebuilding	
	Mean	Std	Mean	Std
Easom (2)	−1.0	0.0	−1.0	0.0
Shubert (2)	−186.03	0.708244	−186.723	0.016331
Rosenbrock (2)	0.0	0.0	0.0	0.0
Zakharov (2)	0.0	0.0	0.0	0.0
De Jong (3)	0.0	0.0	0.0	0.0
Shekel (4, 5)	−5.56783	3.517801	−8.09613	3.003044
Shekel (4, 7)	−9.12732	2.851160	−10.2611	0.440954
Shekel (4, 10)	−7.65202	3.724529	−10.3554	0.966638
Sphere (10)	0.0	0.0	0.0	0.0
Rosenbrock (10)	0.000006	0.000005	0.002511	0.009392
Rastrigin (10)	1.790926	1.294722	0.646338	0.800856
Griewank (10)	0.018083	0.015917	0.01645	0.009914
Zakharov (10)	0.0	0.0	0.0	0.0
Sphere (20)	0.0	0.0	0.0	0.0
Rosenbrock (20)	0.931281	1.685579	0.267050	0.994099
Rastrigin (20)	3.349695	1.807129	2.466102	1.765681
Griewank(20)	0.008904	0.011122	0.007213	0.012479
Zakharov (20)	0.0	0.0	0.0	0.0
Sphere (30)	0.0	0.0	0.0	0.0
Rosenbrock (30)	1.205757	1.838271	0.801094	1.598004
Rastrigin (30)	4.974795	1.973266	4.680690	2.632701
Griewank (30)	0.005735	0.011983	0.003364	0.007581
Zakharov (30)	0.000001	0.000004	0.0	0.0

4.2.5. Worst-Case Analysis

As the proposed CFA is a stochastic optimization algorithm, every single execution of the program may produce a different solution. It thus becomes very important to measure the solution variation of the worst case obtained by the CFA. We conduct the worst-case analysis on the three most challenging functions, namely Rosenbrock (30), Rastrigin (30), and Griewank (30) as follows. One thousand independent runs of the CFA program are performed. We record the worst function value (fitness) that could be obtained by allowing different number of repetitive runs in the experiment. It is seen from Figure 2 that the Rosenbrock (30) fitness value of less than 5 can be obtained with 99.7% confidence because three out of the one thousand runs report a fitness value greater than 5. It also indicates the worst fitness value we could obtain is no more than 5 if we can run the CFA program at least three times. Actually, there are two major components with the fitness distribution. One is close to 4 (a local optimum) with about 20% of the distribution, the other is near zero (the global optimum) with about 79% confidence. The worst-case analysis with Rastrigin (30) is illustrated in Figure 3. We observe a Gaussian-like distribution and the major component falls in the fitness value between 0 and 12. The Gaussian-like distribution provides a reliable guarantee that the mean performance value can be obtained with high confidence. Figure 4 shows the worst-case analysis for Griewank (30). The distribution has a long tail, and the major component concentrates at the high quality part. This situation manifests good properties that there is only one major outcome which is near the global optimum, and that a near-optimal solution can be obtained with a few runs in the worst case.

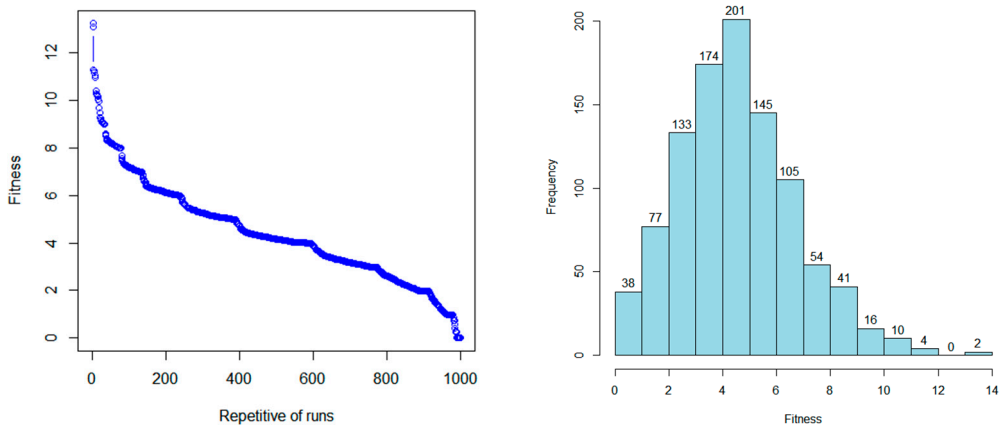
Table 6. The mean best function value and the standard deviation obtained by CFA with or without landscape analysis.

Functions	No Adaptive Landscape Analysis		With Adaptive Landscape Analysis	
	Mean	**Std**	**Mean**	**Std**
Easom (2)	−0.99997	0.000020	−1.0	0.0
Shubert (2)	−185.779	0.963969	−186.723	0.016331
Rosenbrock (2)	0.0	0.0	0.0	0.0
Zakharov (2)	0.0	0.0	0.0	0.0
De Jong (3)	0.0	0.0	0.0	0.0
Shekel (4, 5)	−6.50369	2.613879	−8.09613	3.003044
Shekel (4, 7)	−7.40191	1.802712	−10.2611	0.440954
Shekel (4, 10)	−7.80353	1.033856	−10.3554	0.966638
Sphere (10)	0.0	0.0	0.0	0.0
Rosenbrock (10)	0.797320	1.594630	0.002511	0.009392
Rastrigin (10)	0.734548	0.827677	0.646338	0.800856
Griewank (10)	0.015448	0.011793	0.016450	0.009914
Zakharov (10)	0.0	0.0	0.0	0.0
Sphere (20)	0.0	0.0	0.0	0.0
Rosenbrock (20)	0.797828	1.594400	0.267050	0.994099
Rastrigin (20)	2.590780	1.391706	2.466102	1.765681
Griewank (20)	0.012054	0.013121	0.007213	0.012479
Zakharov (20)	0.0	0.0	0.0	0.0
Sphere (30)	0.0	0.0	0.0	0.0
Rosenbrock (30)	0.801375	1.599876	0.801094	1.598004
Rastrigin (30)	4.704743	1.842029	4.680690	2.632701
Griewank (30)	0.005663	0.007891	0.003364	0.007581
Zakharov (30)	0.0	0.0	0.0	0.0



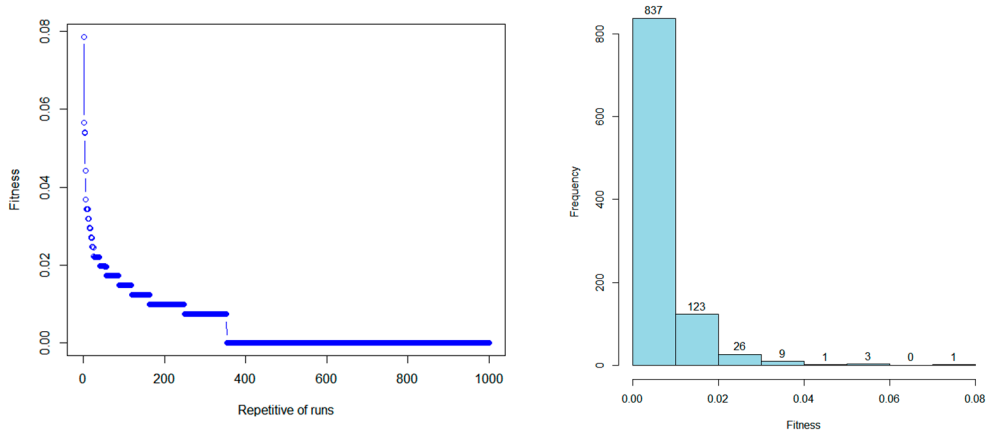
(**a**) (**b**)

Figure 2. Worst-case analysis with the Rosenbrock (30) function. (**a**) The worst fitness obtained by the CFA program as the number of repetitive runs increases. (**b**) The number of program runs with which each performance level is reached.



(**a**) (**b**)

Figure 3. Worst-case analysis with the Rastrigin (30) function. (**a**) The worst fitness obtained by the CFA program as the number of repetitive runs increases. (**b**) The number of program runs with which each performance level is reached.



(**a**) (**b**)

Figure 4. Worst-case analysis with the Griewank (30) function. (**a**) The worst fitness obtained by the CFA program as the number of repetitive runs increases. (**b**) The number of program runs with which each performance level is reached.

4.3. Performance Evaluation

In this section, the performance of the CFA is evaluated in three-fold. First, the CFA is compared against its counterparts, the GSO and the FA. Secondly, the CFA is compared to other kinds of metaheuristics. Thirdly, the performance of CFA is further justified on the CEC 2005 dataset. For the first two experiments, all compared algorithms were executed 30 times for each test function. For each run, the program can perform 160,000 FEs. However, for the third experiment, the evaluation criteria in the original paper are respected where the competing method is executed in 25 independent runs and in each run the method is evaluated at different numbers of consumed FEs. To compare the performance between two competing algorithms, we employ the performance index defined by Yin et al. [6] as follows. Given two competing algorithms, p and q, the performance merit of p against q on a test function is defined by the formula,

$$\text{Merit}(p, q) = (f_p - f^* + \varepsilon)/(f_q - f^* + \varepsilon) \quad (11)$$

where ε is a small constant equal to 5×10^{-7}, f_p and f_q are the mean best function values obtained by the competing algorithms p and q, and f* is the global minimum of the test function. As all the test functions involve minimization, we realize p outperforms q if Merit(p, q) < 1.0, p is inferior to q if Merit(p, q) > 1.0, and p and q perform equally well if Merit(p, q) = 1.0.

4.3.1. Comparison with CFA Counterparts

Our CFA enhances the GSO and the FA by incorporating the responsive strategies from the adaptive memory programming domain. Thus, it is important to validate our idea by comparing the performance of the CFA against its counterparts, the GSO and the FA. The results shown in the second to the fifth columns of Table 7 are the mean best function values obtained by the competing algorithms, and those in the last three columns correspond to the relative merit values. We observe that for the test functions with fewer than ten variables, the CFA has a unit merit or a merit value less than one with one order of magnitude, indicating that all the competing algorithms perform nearly equally well. For the test functions with ten or more variables, the merit value of the CFA in relation to the GSO and the FA becomes significantly less, implying a superior performance in favor of the CFA. The product of the merit values gives an overview of the comparative performance on all test function. It is seen at the bottom of Table 7 that the CFA has a merit product of 7.32×10^{-43} and 9.51×10^{-32} in relation to the GSO and the FA, respectively. We further compare the CFA to the better one of the best function values obtained by the GSO and the FA, giving a merit product of 8.99×10^{-29} as shown in the last column of Table 7. The result suggests that the CFA significantly outperforms its counterparts over the benchmark dataset, and that the CFA also beats a hybrid which takes the better one of the best function values obtained by the two counterparts.

Table 7. The performance comparison between the CFA and its counterparts.

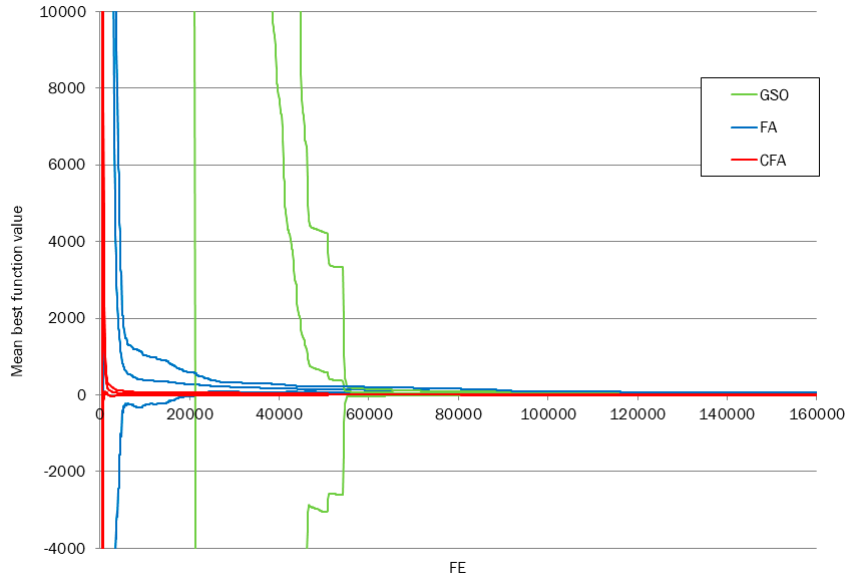
Functions	(a) GSO	(b) FA	(c) Min (a, b)	(d) CFA	Merit (d, a)	Merit (d, b)	Merit (d, c)
Easom (2)	−0.999999	−0.999999	−0.999999	−1.0	1.0	1.0	1.0
Shubert (2)	−184.059055	−91.464488	−184.059055	−186.723	9.86×10^{-1}	4.90×10^{-1}	9.86×10^{-1}
Rosenbrock (2)	0.000007	0.0	0.0	0.0	6.50×10^{-2}	1.0	1.0
Zakharov (2)	0.0	0.0	0.0	0.0	1.0	1.0	1.0
De Jong (3)	0.0	0.0	0.0	0.0	1.0	1.0	1.0
Shekel (4, 5)	−6.883340	−3.339854	−6.883340	−8.09613	8.50×10^{-1}	4.13×10^{-1}	8.50×10^{-1}
Shekel (4, 7)	−6.624701	−2.593236	−6.624701	−10.2611	6.46×10^{-1}	2.53×10^{-1}	6.46×10^{-1}
Shekel (4, 10)	−7.770920	−2.211454	−7.770920	−10.3554	7.50×10^{-1}	2.14×10^{-1}	7.50×10^{-1}
Sphere (10)	0.021677	0.000045	0.000045	0.0	2.31×10^{-5}	1.11×10^{-2}	1.11×10^{-2}
Rosenbrock (10)	1.658934	1.070858	1.070858	0.002511	1.51×10^{-3}	2.35×10^{-3}	2.35×10^{-3}
Rastrigin (10)	2.499270	11.099298	2.499270	0.646338	2.59×10^{-1}	5.82×10^{-2}	2.59×10^{-1}
Griewank (10)	0.154192	17.666979	0.154192	0.01645	1.07×10^{-1}	9.31×10^{-4}	1.07×10^{-1}
Zakharov (10)	0.000330	0.000115	0.000115	0.0	1.51×10^{-3}	4.31×10^{-3}	4.31×10^{-3}
Sphere (20)	0.085764	0.000264	0.000264	0.0	5.83×10^{-6}	1.89×10^{-3}	1.89×10^{-3}
Rosenbrock (20)	8.473378	21.006502	8.473378	0.26705	3.15×10^{-2}	1.27×10^{-2}	3.15×10^{-2}
Rastrigin (20)	5.550878	32.150683	5.550878	2.466102	4.44×10^{-1}	7.67×10^{-2}	4.44×10^{-1}
Griewank (20)	0.553463	0.600160	0.553463	0.007213	1.30×10^{-2}	1.20×10^{-2}	1.30×10^{-2}
Zakharov (20)	0.002574	0.001387	0.001387	0.0	1.94×10^{-4}	3.60×10^{-4}	3.60×10^{-4}
Sphere (30)	0.151057	0.000761	0.000761	0.0	3.31×10^{-6}	6.57×10^{-4}	6.57×10^{-4}
Rosenbrock (30)	23.373028	31.808803	23.373028	0.801094	3.43×10^{-2}	2.52×10^{-2}	3.43×10^{-2}
Rastrigin (30)	9.135285	59.807848	9.135285	4.68069	5.12×10^{-1}	7.83×10^{-2}	5.12×10^{-1}
Griewank (30)	1.053057	0.010732	0.010732	0.003364	3.19×10^{-3}	3.13×10^{-1}	3.13×10^{-1}
Zakharov (30)	2.774662	0.008677	0.008677	0.000001	4.98×10^{-7}	1.59×10^{-4}	1.59×10^{-4}
Merit Product					$\mathbf{7.32 \times 10^{-43}}$	$\mathbf{9.51 \times 10^{-32}}$	$\mathbf{8.99 \times 10^{-29}}$

We further verify the online performance advantage of our CFA against its counterparts by a statistical test suggested in Taillard [28]. As the best objective value produced by a metaheuristic approach is non-deterministic, we model the result obtained from multiple runs of method A (and method B) as a random variable X_a (X_b) and we want to testify the confidence regarding that X_a is less than X_b. A classic statistical test based on the central limit theorem for comparing two proportions is to approximate the mean $X_a - X_b$ as a normal distribution if the collected number of samples

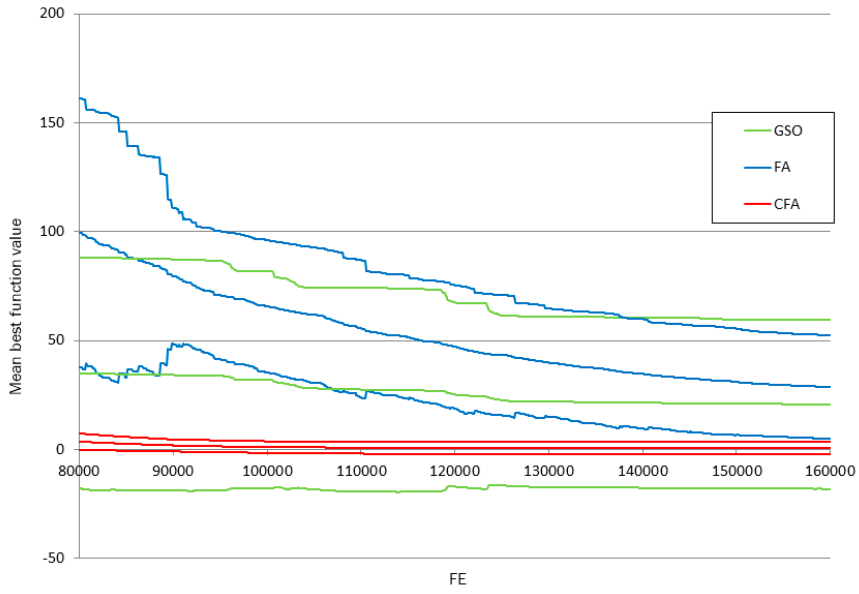
is sufficiently large. Therefore, 30 independent runs of each competing algorithm were conducted. For each run, the online best function value obtained at a particular FE, say e, is a sample for $X_a(e)$, the random variable for the result obtained by method A at FE e. We tally the samples at every instance of FEs during the whole duration of executing the algorithm. By examining the mean curve of $X_a(e)$ and $X_b(e)$ as e increases during the evolution, we can testify if method A well outperforms method B. For clear illustration, the boundary and mean curves over the 30 runs are plotted. Figure 5a shows the online performance analysis with 95% confidence interval for the Rosenbrock (30) function. It is seen that the 95% confidence interval of the best function value obtained by the three competing algorithms (GSO, FA, and CFA) converges with various speeds. Our CFA converges at a much faster speed and reach towards a better function value than its two counterparts. The FA is the second-best performer followed by the GSO. To investigate the detailed performance during the second half duration of the execution, we enlarge the plot for this period as shown in Figure 5b. It can be seen that the CFA significantly outperforms the other two algorithms with 95% confidence level during the second half execution duration. We also found that the GSO performs better than the FA after 80,000 FEs, although the GSO may not beat the FA at the early stage of the execution as previously noted. The online performance comparison with 95% confidence level for the Rastrigin (30) function is shown in Figure 6. We observe that during the whole execution period the CFA significantly outperforms the GSO and the FA. The FA performs better than the GSO when the allowed number of consumed FEs is less than 20,000, but the FA is far surpassed by the GSO if more FEs are allowed. Figure 7 shows the online performance variation with 95% confidence level for the Griewank (30) function. Again, the CFA is the best performer among the three algorithms throughout the whole execution duration. However, we see a phenomenon differing from those for the two previous test functions with the GSO and the FA. The GSO and the FA performs about equally well before consuming 50,000 FEs, although the former is a more stable performer because it has a shorter confidence interval. However, after this critical execution period, the FA becomes very effective both in the convergence speed and the function value. As shown in Figure 7b, the FA significantly surpasses the GSO and reaches a comparative performance with the CFA.

4.3.2. Comparison against Other Metaheuristics

We now compare the CFA against other metaheuristics inspired by different nature metaphors, the PSO, the GA, and the cyber swarm algorithm (CSA). The compared PSO is the constriction factor version proposed by Clerc and Kennedy [29] which has been shown to be one of the best PSO implementations. The implemented GA employs real-value chromosome coding, tournament selection (with $k = 2$, i.e., two competitors in each instance of selection), arithmetic crossover, and Gaussian mutation. The GA is generational without population gap, i.e., the whole parent population is replaced by the offspring population. The implementation and parameter setting of CSA follow the original paper [6]. All the compared algorithms have the same population size of 60 individuals, and are executed until consuming 160,000 FEs. Table 8 tabulates the mean best function value obtained by the compared algorithms over 30 runs and the merit value among the competitors. For the comparison of the CFA against the PSO and the GA, we observe that the CFA well surpasses the other two algorithms on most of the benchmark functions. The merit product in relation to the PSO and the GA is 8.80×10^{-21} and 2.11×10^{-36}, respectively. When we compare the CFA to the CSA, the merit product is 1.94×10^{18}. The result seems to suggest that the CSA performs better on the dataset. However, if we take a closer look, the CSA is very effective in solving small functions with less than ten variables, thus CSA gives significantly greater merits for these functions. For the test functions with ten or more variables, the merit value turns to be in favor to the CFA, disclosing that the CFA is more effective than CSA in tackling larger-sized functions. It is worth noting that both CFA and the CSA take advantage of the features contained in the AMP domain, and the two algorithms extremely outperform the other compared algorithms in our experiments. This phenomenon discloses the potential of future research in the direction of marrying the AMP with other types of metaheuristics.

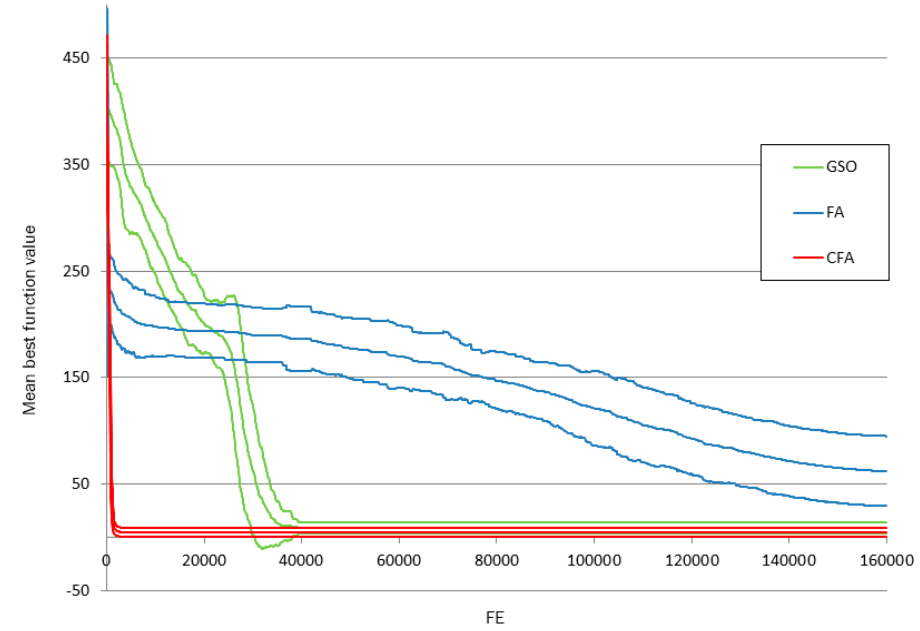


(**a**)

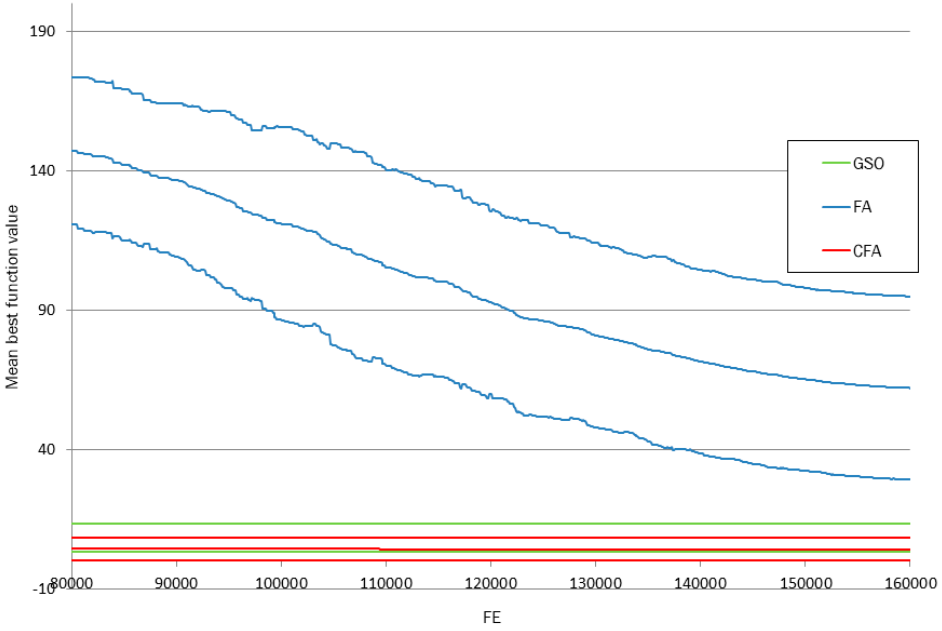


(**b**)

Figure 5. Online performance analysis with 95% confidence interval for the Rosenbrock (30) function. (**a**) The convergence of the best function value during the whole duration of the execution. (**b**) The convergence of the best function value during the second half duration of the execution.

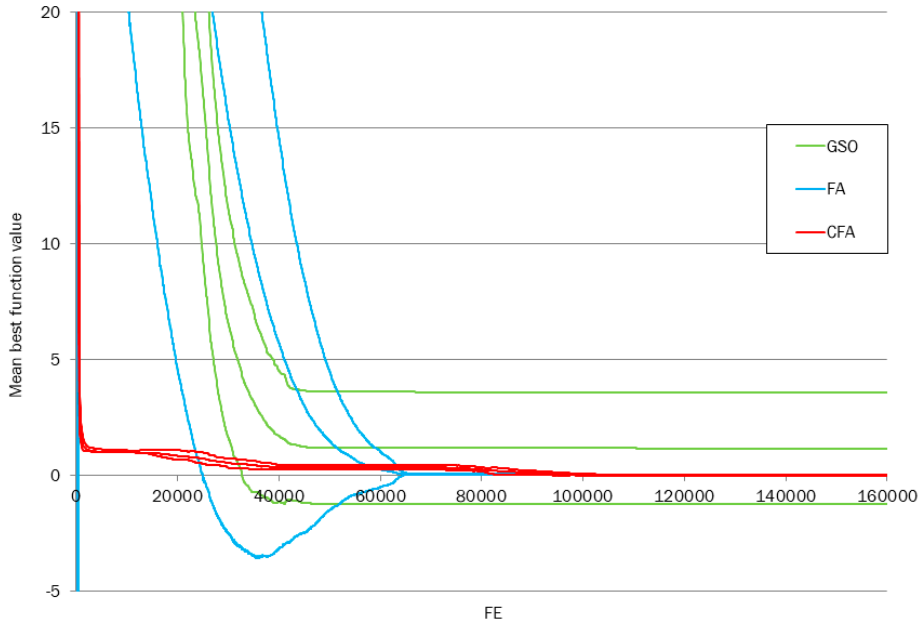


(**a**)

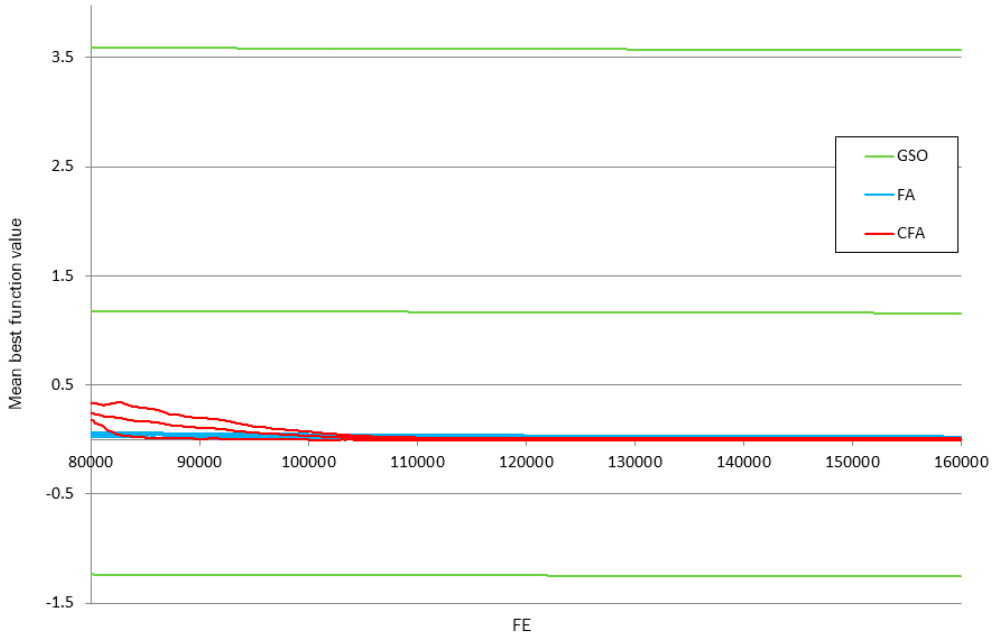


(**b**)

Figure 6. Online performance analysis with 95% confidence interval for the Rastrigin (30) function. (**a**) The convergence of the best function value during the whole duration of the execution. (**b**) The convergence of the best function value during the second half duration of the execution.



(**a**)



(**b**)

Figure 7. Online performance analysis with 95% confidence interval for the Griewank (30) function. (**a**) The convergence of the best function value during the whole duration of the execution. (**b**) The convergence of the best function value during the second half duration of the execution.

Table 8. The performance comparison between the CFA and other metaheuristics.

Functions	(a) PSO	(b) GA	(c) CSA	(d) CFA	Merit (d, a)	Merit (d, b)	Merit (d, c)
Easom (2)	−1.0	−1.0	−1.0	−1.0	1.0	1.0	1.0
Shubert (2)	−186.7309	−186.7309	−186.7309	−186.7232	1.54×10^{4}	1.54×10^{4}	1.54×10^{4}
Rosenbrock (2)	0.0	0.1015	0.0	0.0	1.0	4.93×10^{-6}	1.0
Zakharov (2)	0.0	0.0	0.0	0.0	1.0	1.0	1.0
De Jong (3)	0.0	0.0	0.0	0.0	1.0	1.0	1.0
Shekel (4, 5)	−6.6329	−10.0535	−10.1532	−8.0961	5.84×10^{-1}	2.06×10^{1}	4.11×10^{6}
Shekel (4, 7)	−8.0176	−10.0637	−10.4029	−10.2611	5.94×10^{-2}	4.18×10^{-1}	2.84×10^{5}
Shekel (4, 10)	−7.4195	−10.0750	−10.5364	−10.3554	5.81×10^{-2}	3.92×10^{-1}	3.62×10^{5}
Sphere (10)	0.0	0.0009	0.0	0.0	1.0	5.55×10^{-4}	1.0
Rosenbrock (10)	0.4727	8.1009	0.1595	0.0025	5.29×10^{-3}	3.09×10^{-4}	1.57×10^{-2}
Rastrigin (10)	6.4672	0.0004	0.7464	0.6463	9.99×10^{-2}	1.61×10^{3}	8.66×10^{-1}
Griewank (10)	0.0644	0.0493	0.0474	0.0164	2.55×10^{-1}	3.33×10^{-1}	3.46×10^{-1}
Zakharov (10)	0.0	0.1809	0.0	0.0	1.0	2.76×10^{-6}	1.0
Sphere (20)	0.0	0.0069	0.0	0.0	1.0	7.25×10^{-5}	1.0
Rosenbrock (20)	0.3992	9.0326	0.4788	0.2671	6.69×10^{-1}	2.96×10^{-2}	5.58×10^{-1}
Rastrigin (20)	18.7052	0.0036	6.8868	2.4661	1.32×10^{-1}	6.85×10^{2}	3.58×10^{-1}
Griewank (20)	0.0227	0.0533	0.0128	0.0072	3.17×10^{-1}	1.35×10^{-1}	5.63×10^{-1}
Zakharov (20)	2.683	37.1846	0.0	0.0	1.86×10^{-7}	1.34×10^{-8}	1.0
Sphere (30)	0.0	0.0226	0.0	0.0	1.0	2.21×10^{-5}	1.0
Rosenbrock (30)	9.2089	83.0118	0.3627	0.8011	8.70×10^{-2}	9.65×10^{-3}	2.21
Rastrigin (30)	33.9281	0.0115	11.9425	4.6807	1.38×10^{-1}	4.07×10^{2}	3.92×10^{-1}
Griewank (30)	0.0093	0.0893	0.0052	0.0034	3.66×10^{-1}	3.81×10^{-2}	6.54×10^{-1}
Zakharov (30)	5.4347	136.7129	0.0	0.0	9.20×10^{-8}	3.66×10^{-9}	1.0
Merit Product					$\mathbf{8.80 \times 10^{-21}}$	$\mathbf{2.11 \times 10^{-36}}$	$\mathbf{1.94 \times 10^{18}}$

To compare the CFA with the state-of-the-art variants of FA, we quote the results (mean objective value over 30 runs) from the original paper LFA [9], VESSFA [10], WFA [11], CLFA [12], FAtidal [13], and GDAFA [14]. The best mean objective value for each function obtained by all compared methods is printed in bold. As can be seen in Table 9, our CFA wins the most times as obtaining the best mean objective value among all. GDAFA seems to possess better performance as the dimensionality increases. Both CFA and GDAFA can gain an objective value very close to the optimum, while the other competing methods may produce an objective value far away from the optimum in some challenging functions.

Table 9. The performance comparison between the CFA and the state-of-the-art variants of FA.

Functions	LFA	FAtidal	VSSFA	GDAFA	WFA	CLFA	CFA
Sphere (2)	0.043	**0.0**	–	–	–	–	**0.0**
Rosenbrock (2)	1.34	0.0076	–	–	–	–	**0.0**
Zakharov (2)	0.4950	**0.0**	–	–	–	–	**0.0**
Sphere (10)	17.5	**0.0**	**0.0**	**0.0**	0.0690	**0.0**	**0.0**
Rosenbrock (10)	3.15×10^{4}	9.93	–	–	–	–	**0.0025**
Rastrigin (10)	85.5	7.20	8.6769	**0.0**	10.6039	11.4284	0.6463
Griewank (10)	0.0069	–	**0.0**	**0.0**	0.0075	**0.0**	0.0164
Zakharov (10)	158.0	**0.0**	–	–	–	–	**0.0**
Sphere (30)	14.0818	–	17.0501	1.61×10^{-5}	0.2373	0.1942	**0.0**
Rastrigin (30)	38.3584	–	133.9519	**0.0619**	42.7987	91.6579	4.6807
Griewank (30)	0.4365	–	0.5380	$\mathbf{3.65} \times 10^{-6}$	0.0123	0.0099	0.0034

4.3.3. Comparison on the CEC 2005 Dataset

To further justify the performance of CFA, we compare CFA with the investigated methods reported in [26] on 12 CEC 2005 benchmark functions [30]. The IEEE CEC Repository [27] provides fruitful benchmark datasets for optimization problems with various purposes such as unconstrained, constrained, and multi-objective optimization. The CEC 2005 dataset is designed for unconstrained real-parameter optimization which is addressed in this paper. We selected 12 CEC 2005 functions which are very challenging and have never been solved to optimal by any known methods [26]. We executed

all compared algorithms with the same evaluation criteria and parameter settings as described in the original paper [26]. Each algorithm is executed for 25 independent runs on each test function with n = 10 and 30, respectively. All compared methods are executed by being allowed to consume 1000, 10,000, and 100,000 FEs.

We adopt the GAP performance measure proposed in the original paper [26] and it is defined as GAP = |f − f*| where f is the function value obtained by an evaluated method and f* is the global optimum value of the test function. Table 10 shows the mean minimum (Min.) and average (Avg.) of GAP and Merit of all compared methods over the 12 functions for n = 10. We observe that our CFA is less exploitative in small size CEC problems than the leading methods such as G-CMA-ES and L-CMA-ES, both of which are based on the covariance matrix adaptation evolution strategy (CMA-ES) [31] which updates the covariance of the multivariate distribution to better handle the dependency between variables. Though the CFA is less competitive in the mean Avg. GAP, it can deliver a quality Min. out of 25 independent runs. It suggests that the CFA can be executed multiple times and output the best value from those runs when tackling small size yet complex problems. This phenomenon is also revealed in the geometric mean of the merits (GMM). The GMM of the Min. function value is 1.099, 0.995, and 0.878 at 1000, 10,000, and 100,000 FEs, while the GMM of the Avg. function value gradually deteriorates from 0.963, 1.167 to 1.211 at the same FEs.

Table 10. Min./Avg. GAP and Merit over the CEC dataset with n = 10.

FEs	GAP			Merit		
	1000	**10,000**	**100,000**	**1000**	**10,000**	**100,000**
	Min./Avg.	**Min./Avg.**	**Min./Avg.**	**Min./Avg.**	**Min./Avg.**	**Min./Avg.**
CFA	572.3/751.2	317.6/554.4	212.4/441.1	1.0/1.0	1.0/1.0	1.0/1.0
CSA	382.0/665.9	291.3/506.6	234.5/432.6	1.498/1.128	1.090/1.094	0.906/1.020
STS	616.1/759.4	348.9/576.6	198.3/413.4	0.929/0.989	0.910/0.961	1.071/1.067
G-CMA-ES	269.7/542.0	260.0/419.4	256.0/265.3	2.122/1.386	1.222/1.322	0.830/1.663
EDA	669.9/1059.1	287.1/335.1	269.4/300.6	0.854/0.709	1.106/1.654	0.788/1.467
BLX-MA	456.7/711.1	315.5/445.1	306.2/430.1	1.253/1.056	1.007/1.246	0.694/1.026
SPC-PNX	621.7/750.3	279.6/391.0	206.0/309.9	0.921/1.001	1.136/1.418	1.031/1.423
BLX-GL50	676.0/716.3	272.8/341.0	257.2/319.0	0.847/1.049	1.164/1.626	0.826/1.383
L-CMA-ES	289.0/825.7	225.9/655.8	202.7/411.1	1.980/0.910	1.406/0.845	1.048/1.073
DE	715.4/914.1	396.7/492.4	228.8/272.0	0.800/0.822	0.801/1.126	0.928/1.622
K-PCX	671.0/968.5	488.0/564.4	257.4/475.6	0.853/0.776	0.651/0.982	0.825/0.927
Co-EVO	672.6/799.0	437.5/623.5	268.3/465.4	0.851/0.940	0.726/0.889	0.792/0.948
Merit Product				2.833/0.663	0.949/5.493	0.238/8.195
Geometric Mean				1.099/0.963	0.995/1.167	0.878/1.211

As for high-dimensional and complex CEC problems with n = 30, the mean Min. and Avg. of GAP and Merit of all compared methods are tabulated in Table 11. It is seen from the GMM that our CFA is compared favorably to the other methods in both Min. and Avg. function value at all FEs check points. The prevailing exploration search conducted by CFA is due to its elements of AMP-responsive strategies, which are more effective when the problem is more complex and is presented in higher-dimensional space. The G-CMA-ES is again the best method since it excels in terms of GAP in most cases as compared to the other competing methods. It is worth further studying the possibility of including the CMA technique into the CFA to resolve the dependency between variables.

Table 11. Min./Avg. GAP and Merit over the CEC dataset with $n = 30$.

FEs	GAP			Merit		
	1000	**10,000**	**100,000**	**1000**	**10,000**	**100,000**
	Min./Avg.	**Min./Avg.**	**Min./Avg.**	**Min./Avg.**	**Min./Avg.**	**Min./Avg.**
CFA	792.5/1033.1	450.8/778.6	418.7/602.5	1.0/1.0	1.0/1.0	1.0/1.0
CSA	629.2/785.0	454.4/647.8	420.6/578.2	1.260/1.316	0.992/1.202	0.995/1.042
STS	829.3/957.0	614.9/747.3	431.3/540.3	0.956/1.080	0.733/1.042	0.971/1.115
G-CMA-ES	570.3/658.4	414.4/526.8	405.7/493.0	1.390/1.569	1.088/1.478	1.032/1.222
EDA	39,742/63,491	11,951/26,418	653.6/934.7	0.020/0.016	0.038/0.029	0.641/0.645
BLX-MA	792.9/1198.7	443.9/502.4	410.7/457.2	0.999/0.862	1.016/1.550	1.019/1.318
SPC-PNX	29,793/74,050	637.6/850.1	414.8/430.0	0.027/0.014	0.707/0.916	1.009/1.401
BLX-GL50	8545.4/20,008	474.8/545.9	433.0/507.5	0.093/0.052	0.949/1.426	0.967/1.187
L-CMA-ES	790.8/1009.8	447.6/722.6	404.6/617.0	1.002/1.023	1.007/1.077	1.035/0.976
DE	3473.3/14,461	726.0/781.8	558.7/592.0	0.228/0.071	0.621/0.996	0.749/1.018
K-PCX	27,749/108,623	27,719/108,602	866.1/2257.2	0.029/0.010	0.016/0.007	0.483/0.267
Co-EVO	908.5/1025.8	7496/822.0	625.3/734.5	0.872/1.007	0.060/0.947	0.670/0.820
Merit Product				$4.7 \times 10^{-7}/2.0 \times 10^{-8}$	$1.2 \times 10^{-5}/8.0 \times 10^{-4}$	$1.5 \times 10^{-1}/4.3 \times 10^{-1}$
Geometric Mean				0.266/0.195	0.358/0.523	0.846/0.927

5. Concluding Remarks and Future Research

We have proposed the CFA which is a more effective form of the GSO and the FA in global optimization. The CFA incorporates several AMP strategies including multiple guiding solutions, pattern search as local improvement method, solution set rebuilding in a multi-start search template, and the responsive strategies. The experimental result on benchmark functions for global optimization has shown that the CFA performs significantly better in terms of both solution quality and robustness than the GSO, FA, and several state-of-the-art metaheuristic methods as demonstrated in our statistical analyses and comprehensive experiments. It is worth noting that it is certain a sophisticated method such as CFA incorporating advanced components will pose higher computational complexity in the computation iteration than an algorithm which does not. However, as metaheuristic approaches are computational ones which can stop at any computation iteration and output the best-so-far result. We conduct a fair performance comparison between two metaheuristic approaches at the same number of fitness FE instead of using the evolution iterations. All our experiments follow this fashion.

Our findings strengthen the motivations for marrying the approaches selected from each of the metaheuristic dichotomies, respectively. The CFA template gives general ideas for creating this sort of effective hybrid metaheuristics. Inspired by the promising result of the CFA, it is worthy of investigating the possibility of the application of the CFA template to other metaheuristic approaches with various problem domains for future research.

Author Contributions: Conceptualization, P.-Y.Y.; methodology, P.-Y.Y. and P.-Y.C.; software, P.-Y.C.; validation, Y.-C.W. and R.-F.D.; writing—original draft preparation, P.-Y.Y.; writing—review and editing, P.-Y.Y., Y.-C.W. and R.-F.D.; visualization, P.-Y.C.; funding acquisition, P.-Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by MOST Taiwan, grant numbers 107-2410-H-260-015-MY3. The APC was funded by 107-2410-H-260-015-MY3.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Nomenclature

The list of the acronyms referenced in this paper is tabulated as follows.

EA	evolutionary algorithm
GSO	glowworm swarm optimization
FA	firefly algorithm
AMP	adaptive memory programming
CSA	cyber swarm algorithm
CFA	Cyber Firefly Algorithm
GRASP	greedy randomized adaptive search procedures
GA	genetic algorithm
PSO	particle swarm optimization
VNS	variable neighborhood search
SS	scatter search
SS/PR	path relinking
FDC	fitness distance correlation
FE	function evaluations
CMA-ES	covariance matrix adaptation evolution strategy
GMM	geometric mean of the merits

References

1. Yin, P.Y. *Towards more effective metaheuristic computing, In Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends*; IGI-Global Publishing: Hershey, PA, USA, 2012.
2. Talbi, E.G.; Bachelet, V. COSEARCH: A parallel cooperative metaheuristic. *J. Math. Model. Algorithms* **2006**, *5*, 5–22. [CrossRef]
3. Shen, Q.; Shi, W.M.; Kong, W. Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data. *Comput. Biol. Chem.* **2008**, *32*, 52–59. [CrossRef] [PubMed]
4. Marinakis, Y.; Marinaki, M.; Doumpos, M.; Matsatsinis, N.F.; Zopounidis, C. A hybrid ACO-GRASP algorithm for clustering analysis. *Ann. Oper. Res.* **2011**, *188*, 343–358. [CrossRef]
5. Fuksz, L.; Pop, P.C. A hybrid genetic algorithm with variable neighborhood search approach to the number partitioning problem. *Lect. Notes Comput. Sci.* **2013**, *8073*, 649–658.
6. Yin, P.Y.; Glover, F.; Laguna, M.; Zhu, J.S. Cyber swarm algorithms: Improving particle swarm optimization using adaptive memory strategies. *Eur. J. Oper. Res.* **2010**, *201*, 377–389. [CrossRef]
7. Krishnanand, K.N.; Ghose, D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In Proceedings of the IEEE Swarm Intelligence Symposium, Pasadena, CA, USA, 8–10 June 2005; pp. 84–91.
8. Yang, X.S. Firefly algorithm. *Nat. Inspired Metaheuristic Algorithms* **2008**, *20*, 79–90.
9. Yang, X.S. Firefly algorithm, levy flights and global optimization. In *Research and Development in Intelligent Systems XXVI*; Springer: London, UK, 2010; pp. 209–218.
10. Yu, S.; Zhu, S.; Ma, Y.; Mao, D. A variable step size firefly algorithm for numerical optimization. *Appl. Math. Comput.* **2015**, *263*, 214–220. [CrossRef]
11. Zhu, Q.G.; Xiao, Y.K.; Chen, W.D.; Ni, C.X.; Chen, Y. Research on the improved mobile robot localization approach based on firefly algorithm. *Chin. J. Sci. Instrum.* **2016**, *37*, 323–329.
12. Kaveh, A.; Javadi, S.M. Chaos-based firefly algorithms for optimization of cyclically large-size braced steel domes with multiple frequency constraints. *Comput. Struct.* **2019**, *214*, 28–39. [CrossRef]
13. Yelghi, A.; Köse, C. A modified firefly algorithm for global minimum optimization. *Appl. Soft Comput.* **2018**, *62*, 29–44. [CrossRef]
14. Liu, J.; Mao, Y.; Liu, X.; Li, Y. A dynamic adaptive firefly algorithm with globally orientation. *Math. Comput. Simul.* **2020**, *174*, 76–101. [CrossRef]
15. Wang, J.; Song, F.; Yin, A.; Chen, H. Firefly algorithm based on dynamic step change strategy. In *Machine Learning for Cyber Security*; Chen, X., Yan, H., Yan, Q., Zhang, X., Eds.; Lecture Notes in Computer Science 12487; Springer: Cham, Switzerland, 2020. [CrossRef]
16. Glover, F. Tabu search and adaptive memory programming—Advances, applications and challenges. In *Interfaces in Computer Science and Operations Research*; Kluwer Academic Publishers: London UK, 1996; pp. 1–75.
17. Glover, F. A template for scatter search and path relinking. *Lect. Notes Comput. Sci.* **1998**, *1363*, 13–54.
18. Laguna, M.; Marti, R. *Scatter Search: Methodology and Implementation in C*; Kluwer Academic Publishers: London, UK, 2003.
19. Chen, X.S.; Ong, Y.S.; Lim, M.H.; Tan, K.C. A Multi-Facet Survey on Memetic Computation. *IEEE Trans. Evol. Comput.* **2011**, *15*, 591–607. [CrossRef]
20. Feo, T.A.; Resende, M.G.C. Greedy randomized adaptive search procedures. *J. Glob. Optim.* **1995**, *6*, 109–133. [CrossRef]
21. Hooke, R.; Jeeves, T.A. Direct search solution of numerical and statistical problems. *J. Assoc. Comput. Mach.* **1961**, *8*, 212–229. [CrossRef]
22. Dolan, E.D.; Lewis, R.M.; Torczon, V.J. On the local convergence of pattern search. *Siam J. Optim.* **2003**, *14*, 567–583. [CrossRef]
23. Jones, T.; Forrest, S. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Proceedings of the International Conference on Genetic Algorithms, Morgan Laufman, Santa Fe, NM, USA, 15–19 July 1995; pp. 184–192.
24. Hedar, A.R.; Fukushima, M. Tabu search directed by direct search methods for nonlinear global optimization. *Eur. J. Oper. Res.* **2006**, *170*, 329–349. [CrossRef]

25. Hirsch, M.J.; Meneses, C.N.; Pardalos, P.M.; Resende, M.G.C. Global optimization by continuous GRASP. *Optim. Lett.* **2007**, *1*, 201–212. [CrossRef]
26. Duarte, A.; Marti, R.; Glover, F.; Gortazar, F. Hybrid scatter-tabu search for unconstrained global optimization. *Ann. Oper. Res.* **2011**, *183*, 95–123. [CrossRef]
27. Al-Roomi, A.R. *IEEE Congresses on Evolutionary Computation Repository*; Dalhousie University, Electrical and Computer Engineering: Halifax, NS, Canada, 2015; Available online: https://www.al-roomi.org/benchmarks/cec-database (accessed on 15 November 2020).
28. Taillard, E.D.; Waelti, P.; Zuber, J. Few statistical tests for proportions comparison. *Eur. J. Oper. Res.* **2008**, *185*, 1336–1350. [CrossRef]
29. Clerc, M.; Kennedy, J. The particle swarm explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
30. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.P.; Auger, A.; Tiwari, S. *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*; Technical Report; Nanyang Technology University of Singapore: Singapore, 2005.
31. Hansen, N. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1769–1776.