mathematics

MDPI

Article

An Enhanced Partial Search to Particle Swarm Optimization for Unconstrained Optimization

Shu-Kai S. Fan [1,*] and Chih-Hung Jen [2]

[1] Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei City 10608, Taiwan
[2] Department of Information Management, Lunghwa University of Science and Technology, Guishan, Taoyuan County 33306, Taiwan; f7815@mail.lhu.edu.tw
* Correspondence: morrisfan@ntut.edu.tw

Received: 18 February 2019; Accepted: 9 April 2019; Published: 17 April 2019

Abstract: Particle swarm optimization (PSO) is a population-based optimization technique that has been applied extensively to a wide range of engineering problems. This paper proposes a variation of the original PSO algorithm for unconstrained optimization, dubbed the enhanced partial search particle swarm optimizer (EPS-PSO), using the idea of cooperative multiple swarms in an attempt to improve the convergence and efficiency of the original PSO algorithm. The cooperative searching strategy is particularly devised to prevent the particles from being trapped into the local optimal solutions and tries to locate the global optimal solution efficiently. The effectiveness of the proposed algorithm is verified through the simulation study where the EPS-PSO algorithm is compared to a variety of exiting "cooperative" PSO algorithms in terms of noted benchmark functions.

Keywords: particle swarm optimization (PSO); multiple swarms; cooperative search

1. Introduction

The development of evolutionary computation is motivated by natural evolution. In a population of individuals employed to encode potential solutions of the problem, the evolution is emulated pertaining to the rule of "survival of the fittest", and a better solution is then expected for generation to generation, such as genetic algorithm [1]. With contrast to evolutionary computation, Eberhart and Kennedy [2] developed a new population-based optimization technique, termed particle swarm optimization (PSO). PSO is a stochastic optimization algorithm that operates on a population of a set of initial solutions to explore the search space in a continuous domain. Akin to ant colony optimization [3], the idea of PSO is inspired by the interesting concept of a social behavior, communications and interactions in bird flocking and fish schooling. It has been widely recognized in the literature that PSO has demonstrated consistently good performance in solving various real-valued optimization problems. Even so, the ordinary PSO still suffers a serious problem that it, sometimes, fails to efficiently explore the local neighborhood of the found solution and accurately anchor the optimum. Many previous studies showed that PSO has an innate global search ability, but local search ability can vary mostly from case to case. Shi and Eberhart [4] introduced a time decreasing inertia factor to balance between global and local search ability of the swarm. The hybridization of PSO has been a popular topic in recent days. For instance, the SAPSO [5] is an optimization algorithm combining the PSO with the SA algorithm (Simulated Annealing), where the PSO is used for the global search and SA for the local search. Cui, Zeng and Cai [6] presented the SPSO where the PSO was combined with the Tabu technique to enhance the local search capability of PSO. Zahara, Fan and Tsai [7] presented the PSO hybridized with the Nelder–Mead simplex search method (NM–PSO) to solve the objective of Gaussian

curve fitting and the Otsu's method in the field of image thresholding. The algorithmic development of NM-PSO was fully addressed in Fan, Liang and Zahara [8] and Fan and Zahara [9].

The previously reviewed algorithms have been shown quite effective to improve PSO's local search ability by hybridization. Nonetheless, the PSO is still vulnerable to a risk of falling into a local optimum as the dimensionality of the search space dramatically increases. On this account, various modifications of PSO were developed, trying to make search diversified as to locate the global optimum of high-dimensional problems. Among, one of the most noted cooperative PSO algorithms is the CPSO addressed by van den Bergh and Engelbrecht [10]. The search is conducted by using multiple swarms to optimize different components of the solution vector cooperatively. In addition, the evolutionary operators such as selection, crossover and mutation have been used in the PSO to keep the best particle mobilized, and also to improve the ability to escape from local minima [11,12]. In order to maintain the diversity and to jump from local optima, relocating the particles can be a promising strategy when most of the particles are too close to each other [13]. In their modified PSO, the search space is partitioned into several lower dimensional subspace and multiple swarms are applied to perform a cooperative search. On the other hand, Rada-Vilela et al. [14] proposed the hybrid PSO algorithms that incorporate noise mitigation mechanisms. The performance of the algorithms was analyzed by means of a set of population statistics that measure different characteristics of the swarms throughout the search process. Taghiyeh and Xu [15] proposed the algorithm that works with a set of statistically global best positions including one or more positions with objective function values of statistical equivalence. That PSO algorithm is also integrated with adaptive resampling procedures in order to enhance the capability of coping with noisy objective functions. In this paper, we develop an enhanced PSO termed enhanced partial search particle swarm optimization (EPS-PSO) by using a multi-swarm strategy in attempt to improve the ordinary PSO search that may easily get trapped in a local optimum.

2. Particle Swarm Optimization

From a practical point of view, the PSO is easy to implement and has been shown to perform well on abundant industrial optimization problems. In this section, the original PSO is first introduced, and various up-to-date researches are also reviewed that focus on parameter analysis and modifications of PSO.

2.1. Traditional PSO Operation

In the ordinary PSO, particles are evolved by cooperation among the individuals themselves; hence it can be classified as a population-based technique, where the entire population is called swarm. Each particle represents a possible solution to the optimization problem, and these particles adjust its flying trajectory according to its own flying experience and other particles' flying experience. In other words, each particle is treated as a point in a multi-dimensional space. In iteration, each particle moves in the direction of its own personal best solution visited so far, as well as in the direction of the global best solution discovered so far in the swarm. Pertaining to this kind of social interaction, if a particle discovers a promising new solution that is better than current one, then all the other particles will move toward it, by this way, exploring the region more thoroughly in the search domain. In the PSO, if s denotes the swarm (population) size, then each individual $1 \leq i \leq s$ has the following attributes. The $x_{i,d}$ and $v_{i,d}$ respectively represent the current position and velocity of the i-th particle in the $d-$th dimension. The previous best position of each particle is recorded and represented as $P_{p,d}$, and $P_{g,d}$ as the global best position of all the particles in the swarm. By the previous definitions, the velocity and position update of each particle is performed according to:

$$v_{i,d} = v_{i,d} + c_1 r_1 (P_{p,d} - x_{i,d}) + c_2 r_2 (P_{g,d} - x_{i,d}), \tag{1}$$

$$x_{i,d} = x_{i,d} + v_{i,d}, \tag{2}$$

where c_1 and c_2 denote the acceleration coefficients reflecting the weighting of stochastic acceleration terms that pull each particle toward $P_{p,d}$ and $P_{g,d}$ positions respectively, as well as r_1 and r_2 are two random numbers sampled from the range (0, 1). When updating the velocity of a particle using (1), different dimensions have different random numbers r_1 and r_2. Some research works set the two values equal; by this way, this version of PSO has a limited search space due to the unique random numbers being used for all dimensions. The value of each component in every $v_{i,d}$ vector can be clamped to the range $[-v_{\max}, v_{\max}]$ to reduce the likelihood of particles leaving the search domain. The value of $v_{\max}$ is usually chosen to be $k \times x_{\max}$ with $0.1 \leq k \leq 1.0$ (Eberhart, Dobbins and Simpson [16]). Note that this does not restrict the values of x_i to fall within the range $[-v_{\max}, v_{\max}]$, but only limits "the maximum movement distance" of a particle during each iteration.

A variety of papers have worked on improving the original PSO's performance in different ways, and one of the most well-known modification was the one proposed by Shi and Eberhart [4] that introduced a special parameter called the inertia weight ω into the original PSO as follows:

$$v_{i,d} = \omega v_{i,d} + c_1 r_1 (P_{p,d} - x_{i,d}) + c_2 r_2 (P_{g,d} - x_{i,d}), \tag{3}$$

The choice of the inertia weight can be judiciously made along with a proper selection of the maximum movement distance to optimize the contribution of global and local exploration capabilities. Moreover, Shi and Eberhart [4] observed that a reasonable choice for ω should decrease with a large value of $v_{\max}$. As a general remark, they theorized that a better performance would be obtained if the inertia weight were chosen a time varying, linearly decreasing quantity, rather than being a constant value and supported their statement with a single case study. It was inferred that the PSO search should start with a high inertia weight for coarse global exploration and the inertia weight should linearly decrease to facilitate finer local explorations in later iterations. This should help the search route to approach the optimum of the optimization problem quickly.

2.2. Cooperative Multiple Swarms

Most population-based algorithms attempt to optimize a problem through a single population. In a single population, each individual as agents interact by communicating information to each other while solving a problem. The information exchanged among agents may be misleading if the entire population has been attracted toward the local optima, and should sometimes alter the behavior of the agent receiving it. Therefore, the evolutionary procedure can be as a cooperative learner. With this cooperative conception, applying multiple swarms to execute different search assignment and exchanging the information between swarms appears to be promising. In order to evolve more and more complex landscapes, the cooperative multi-swarm idea has been implemented in the context of genetic algorithms (Cooperative Coevolutionary Genetic Algorithms, CCGAs) by Potter and De Jong [17]. Instead of optimizing the entire solution space simultaneously, the variables were treated individually and optimized by their assigned subpopulations. In doing so, each subpopulation is to optimize an individual variable of the solution vector, then reducing to the multi-tasking of one-dimensional optimization problems. To maintain the feasibility, all the subpopulations need to cooperate each other by information exchange to generate a valid solution vector. The use of multiple interacting subpopulations has also been applied to the PSO, creating a family of CPSOs [10]. In the CPSO, the solution vector was first split into n parts, each part being optimized by a swarm with its m particles. It indicates that each swarm only needs to exert its search capability on n dimensions of the original problem, and individual current best solutions found by every swarm are shared to evolve a unique global best position.

3. Enhanced Partial Search Particle Swarm Optimization (EPS-PSO) Algorithm

In the standard PSO, there always exists a phenomenon that it might lead to a stagnant state if the global best position cannot be improved in several consecutive iterations. In van den Bergh [18], a premature convergence of the standard PSO algorithm has been addressed. To avoid this pitfall, a supplementary search direction may be well supplied as to help the swarm jump out of the local optimum. In Figure 1, an example illustrates a scenario of how the standard PSO should be improved for escaping from the local optimum. The figure displays a contour map of a two-dimensional problem having multiple local optima. The points G1 and G2 are local optima and the point G3 is the global optimum. If the current global best solution/position is very close to G1, then the swarm may be entirely contracted toward G1 and has difficulty to escape from the local region. If the point G2 that exhibits a better solution than the point G1 appears during the optimization steps, then information exchange between these two local regions must rejuvenate the swarm being stagnated. As such, a new search direction (or path) will be developed from G1 to G2. This type of local partial search, if regularly changed in time, will definitely increase the chance of anchoring the global optimum, like G3 in Figure 1.

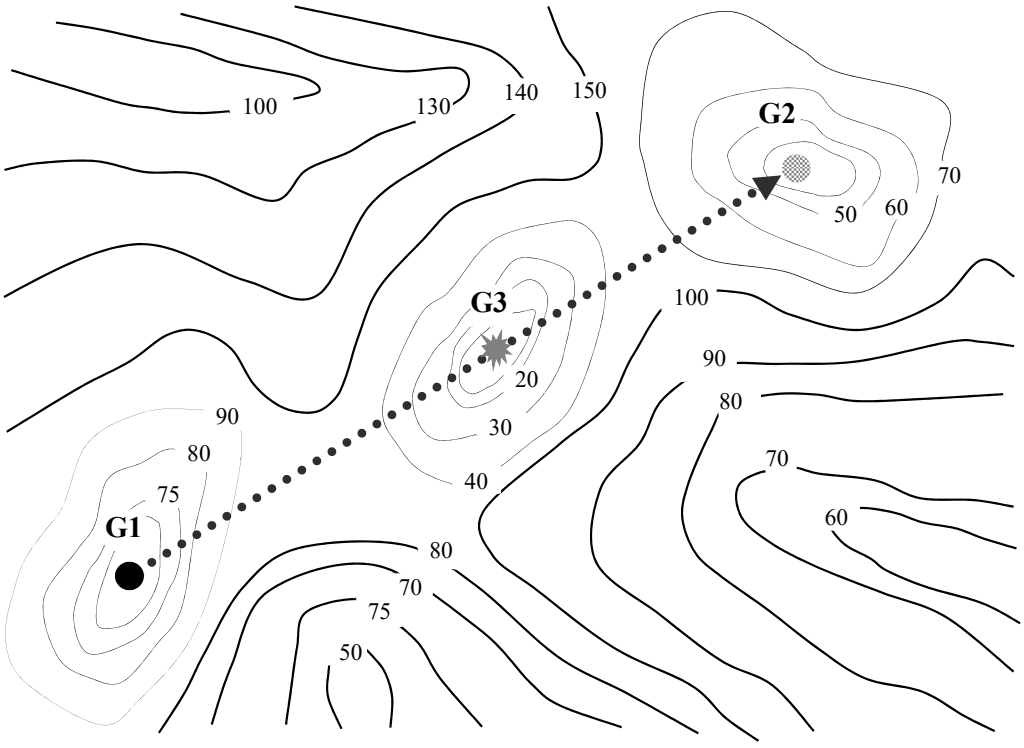


Figure 1. Diagram illustrating how the EPS-PSO algorithm assists the standard PSO in escaping from local optima.

To achieve the foregoing cooperative strategy for unconstrained optimization, an additional population is required to take up the local search assignment. The additional population can be regarded as an enhanced local explorer in addition to the ordinary PSO search, hence the new algorithm to be proposed is termed the Enhanced Partial Search-Particle Swarm Optimization (EPS-PSO) algorithm. To start the EPS-PSO algorithm, first, the entire population is divided into two equal-sized sub-swarms, named the traditional swarm and the co-search swarm, respectively. Every t generations (or iterations), where t is called the re-initialization period, the EPS-PSO algorithm will re-initialize the co-search swarm. There is one exception if the current global best position of the co-search swarm outperforms that of the traditional swarm, then the re-initialization of the co-search swarm will be called off. In other words, the primary difference between the traditional swarm and the co-search

swarm is the search topology assigned. The traditional swarm is mainly aimed at global exploration in the entire search space. By contrast, the co-search swarm is re-initialized periodically for local exploration. In the context of unconstrained optimization, the search space is restrained within the box constraints. Note also that the enhanced partial search is done within the given solution domain and the cooperation/communication occurs only as the co-search swarm exploits better outcomes than the traditional swarm. In a minimization case, let $P_{co-g,d}$ and $P_{T-g,d}$ be the global best positions of the co-search swarm and the traditional swarm, respectively. If the fitness improves by the co-search swarm, $f(P_{co-g,d}) < f(P_{T-g,d})$, then $P_{co-g,d}$ replaces $P_{T-g,d}$ to be the global best position of the traditional swarm (see the illustration in Figure 2b). Otherwise, both swarms proceed on their own without any communication (see the illustration in Figure 2a). The co-search swam provides a new search path to assists the particles of the traditional swarm to jump out of the local optimum, implying that the favorable search information is periodically supplied if appropriate.

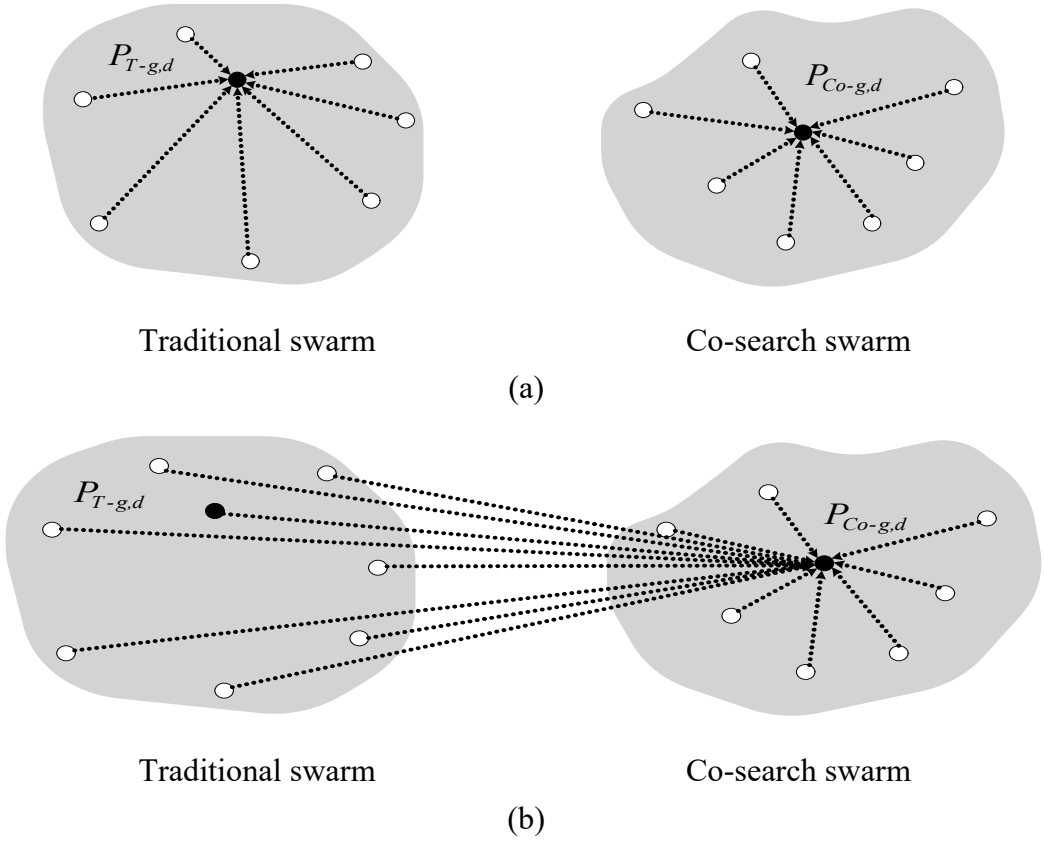


Figure 2. Diagram illustrating the two swarms' convergence situations: (**a**) If $f(P_{Co-g,d}) \geq f(P_{T-g,d})$, (**b**) If $f(P_{Co-g,d}) < f(P_{T-g,d})$.

For the clarity of presentation, the procedure of the EPS-PSO Algorithm 1 can be formalized as below:

The Enhanced Partial Search Particle Swarm Optimization (EPS-PSO) Algorithm 1

Define
m: Each swarm's population size
n: Swarm ID number
t: Re-initialization period
k: function evaluation index
For each particle in each swarm
    Circumscribe the search space for the traditional and co-search swarms within the box constraints
    Initialize position x_i, particle's personal best p_i and velocity v_i for both swarms
    Perform the function evaluation for each particle and update k,
Endfor
Repeat:
For each swarm $j \in [1 \dots n]$:
For each particle $i \in [1 \dots m]$:
        If $f(\boldsymbol{x}_i) < f(\boldsymbol{p}_i)$
        then $\boldsymbol{p}_i \leftarrow \boldsymbol{x}_i$
        If $f(\boldsymbol{p}_i) < f(\boldsymbol{p}_{g,d})$
        then $\boldsymbol{p}_{g,d} \leftarrow \boldsymbol{p}_i$
Endfor
Perform particle velocity and position updates via Equations (1–2)
Endfor
Perform the function evaluation for both swarms and update k
If the criterion of the re-initialization period t for the co-search swarm is met
    For each particle in the co-search swarm
        Circumscribe the search space within the box constraints
            Re-initialize position x_i, particle's personal best p_i and velocity v_i for the co-search swarm
            Perform the function evaluation for each particle and update k
    Endfor
    If $f(P_{co-g,d}) < f(P_{T-g,d})$ **then** $P_{T-g,d} \leftarrow P_{co-g,d}$
Until the maximum number k of function evaluations is satisfied

4. Experiment Setup

To evaluate the performance of the EPS-PSO algorithm, a suite of benchmark functions with different difficulties are chosen. All the functions presented here have the objective value 0 in their global minima. The definitions of each test problem are tabulated in Table 1. Among the five benchmark functions, Rosenbrock and Quadric are unimodal functions while the others are multimodal (i.e., Ackley, Rastrigin and Griewank functions). However, Shang and Qiu [19] had verified that the n-dimensional ($n = 4 \sim 30$) Rosenbrock function is not unimodal and has two minima, one with the optimal objective of zero and the other with the optimal objective of around 3.7~3.9. The performance of the EPS-PSO algorithm will be assessed by these five 30-dimensional bench mark functions $f_1 \sim f_5$, and then the comparison is made against the other two algorithms: the traditional PSO [2] and 4 different versions of CPSO [10]. The CPSO algorithms are briefly mentioned as follows:

- CPSO-S: A maximally "split" swarm where the search space vector is split into 30 parts.
- CPSO-S_6: The search space vector for CPSO-S_6 is split into only six parts (of five components each).
- CPSO-H: A hybrid swarm, consisting of a maximally split swarm, coupled with a plain swarm.
- CPSO-H_6: A hybrid swarm, consisting of a CPSO-S_6 swarm, coupled with a plain swarm.

Table 1. The definition of the benchmark functions.

f_i.	Problems	n	Search dom.	Objective Functions	Global min.
f_1	Rosenbrock	30	$x_i \in [-2.048, 2.048]$	$f_1(x) = \sum_{i=1}^{n/2}\left(100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right)$	$x_i \in 1$; $f(\boldsymbol{x}) = 0$
f_2	Quadratic	30	$x_i \in [-100, 100]$	$f_2(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	$x_i \in 0$; $f(\boldsymbol{x}) = 0$
f_3	Ackley	30	$x_i \in [-30, 30]$	$f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$x_i \in 0$; $f(\boldsymbol{x}) = 0$
f_4	Rastrigin	30	$x_i \in [-5.12, 5.12]$	$f_4(x) = \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$x_i \in 0$; $f(\boldsymbol{x}) = 0$
f_5	Griewank	30	$x_i \in [-600, 600]$	$f_5(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$x_i \in 0$; $f(\boldsymbol{x}) = 0$

All the experiments were conducted using three different swarm sizes of 10, 15 and 20 except the EPS-PSO algorithm that used the population size of 20 (i.e., the sub-swarm size of 10 is used for the traditional swarm and the co-search swarm each). Every algorithm was halted after 2×10^5 function evaluations. Regarding the replication and comparison of computational experiments in applied evolutionary computing, interested readers can refer to the work by Črepinšek et al. [20]. Note again that all the benchmark functions have the same problem dimension of 30. For all the PSO-based algorithms compared in the experimental study, the two random numbers, r_1 and r_2 in (1), were randomly generated from the range $(0, 1)$. The value of $v_{\max}$ is clamped to the allowable range of x_i, and $c_1 = c_2 = 1.49$. In all the implementations, the inertia weight parameter ω decreases with the number of iteration, akin to the scheme described in Shi and Eberhart [21] except the traditional PSO. That is, ω decreases linearly over iteration from 1 down to 0, and the traditional PSO using $\omega = 0.72$ [4]. Moreover, to pose a more challenging optimization task, all the algorithms were further tested using the transformed benchmark functions using Salomon's coordinate rotation [22]. Prior to each individual optimization, a new rotation was independently executed, and therefore no bias was introduced because of a specific rotation. In the co-search swarm, the size of limited search space is set to half the allowable range of each dimension.

Note that all the tested cooperative PSO variants and the proposed EPS-PSO use the same initialization procedure, i.e., the random generation within the box constraints. In the proposed EPS-PSO algorithm, two sub-swarms, traditional and co-search ones, are initialized by random generation. Since then, the co-search swarm is to be re-initialized every re-initialization period only if the co-search swarm is not able to locate a better global best solution than the traditional swarm. If the co-search swarm exploits better outcomes than the traditional swarm, then the re-initialization will not be executed. The proposed EPS-PSO algorithm will be compared to the other 5 PSO variants in terms of the unrotated and rotated versions of the five benchmark functions.

5. Computational Result and Discussion

In this section, the experimental results of five benchmark functions obtained by running 6 different versions of PSO (i.e., including the traditional PSO, 4 different versions of CPSO, and EPS-PSO) are presented. To conduct fair comparisons among algorithms, all experiments were run 50 times, and the computational results are exhibited in Tables 2–6. In the tables, the second column lists the entire population size for the PSOs. The third column lists how many iterations the co-search swarm would be re-initialized in a limited search space for the EPS-PSO algorithm. Five levels of the re-initialization period are examined, t = 100, 500, 1000, 5000, 10,000. As t = 10,000 is used, indicating two traditional swarms of 10 particles each without communication throughout the search process.

The use of $t = 1000$ stands for 10 re-initializations and communications. The fourth and fifth columns list the mean function error with 95% confidence interval after 2×10^5 function evaluations for the unrotated and rotated versions of the benchmark functions. Note that a "new" rotation is performed prior to every individual optimization, so each rotated function has a different functional form but the same global minimum. Thus, the minimum objective value is still zero.

In evolutionary computation, the Rosenbrock function is frequently chosen to evaluate the performance of optimization algorithms due to its highly nonlinear and non-convex properties. The global minimum hides within an elongated valley, as known as "Banana" function. To reach the valley is trivial but to locate accurately the global minimum is of primary importance. In Shang and Qiu [19], the function has been shown to own multiple local minima as the problem size exceeds 4. Table 2 shows that the Rosenbrock function in its unrotated form can be easily solved by the EPS-PSO algorithm as the re-initialization period is $t \geq 500$. The other 5 algorithms cannot compete with the EPS-PSO algorithm. However, when the search space is rotated, the quality of solutions generated by the EPS-PSO algorithm has deteriorated quickly but the other 5 algorithms seem invariant to the coordinate rotation. Even so, the EPS-PSO algorithm still takes the lead. Figure 3 shows the best convergence result of the algorithms among 50 independent runs in terms of the logarithm of function error over iteration. For the un-rotated case, the EPS-PSO algorithm has achieved the global optimum after 1.2×10^5 function evaluations. For the rotated case, the EPS-PSO algorithm dominates the other algorithms after 9×10^4 function evaluations. Note that the CPSO result shown in Figure 3 corresponds to the best convergence result among 4 versions of the CPSO algorithms over 50 independent runs. For further details about the family of CPSOs, interested readers can refer to van den Bergh and Engelbrecht [10].

Table 2. Computational results of Rosenbrock obtained by using 6 PSO-based algorithms after 2×10^5 function evaluations, averaged over 50 independent runs.

Algorithm	s	t	Mean(Unrotated)	Mean(Rotated)
PSO	10	—	2.10e–01 ± 2.61e–01	2.12e–01 ± 7.12e–01
	15	—	1.53e–02 ± 3.31e–02	1.04e–01 ± 6.82e–01
	20	—	4.52e–03 ± 6.18e–03	2.16e–01 ± 2.41e–01
CPSO-S	10	—	6.06e–01 ± 4.61e–01	2.21e+00 ± 6.78e–01
	15	—	3. 63e–01 ± 1.04e–02	1.42e+00 ± 2.23e–01
	20	—	8.16e–01 ± 2.47e–02	4.71e+00 ± 7.50e–01
CPSO-H	10	—	6.11e–01 ± 2.13e–02	7.16e–01 ± 6.88e–01
	15	—	1.14e–02 ± 2.04e–02	2.92e–01 ± 1.13e–01
	20	—	1.15e–01 ± 1.48e–01	2.59e+00 ± 2.06e–01
CPSO-S_6	10	—	8.33e+00 ± 4.81e–01	6.38e+00 ± 9.48e–01
	15	—	1.07e+00 ± 7.26e–01	1.42e+00 ± 9.71e–01
	20	—	6.29e–01 ± 5.03e–01	8.51e+00 ± 3.38e–01
CPSO-H_6	10	—	2.94e–01 ± 2.11e–01	2.17e–01 ± 8.32e–01
	15	—	7.59e–01 ± 5.72e–01	7.24e–01 ± 7.13e–01
	20	—	8.31e–01 ± 6.11e–01	1.13e–01 ± 6.05e–01
EPS-PSO	20	100	2.92e–03 ± 1.02e–03	7.72e–03 ± 1.02e–02
	20	500	2.58e–19 ± 5.28e–19	5.43e–05 ± 3.30e–05
	20	1000	2.90e–22 ± 1.71e–22	9.60e–03 ± 7.24e–03
	20	5000	2.54e–28 ± 1.93e–29	1.64e–03 ± 1.76e–04
	20	10000	1.89e–20 ± 5.01e–20	1.25e–04 ± 3.00e–04

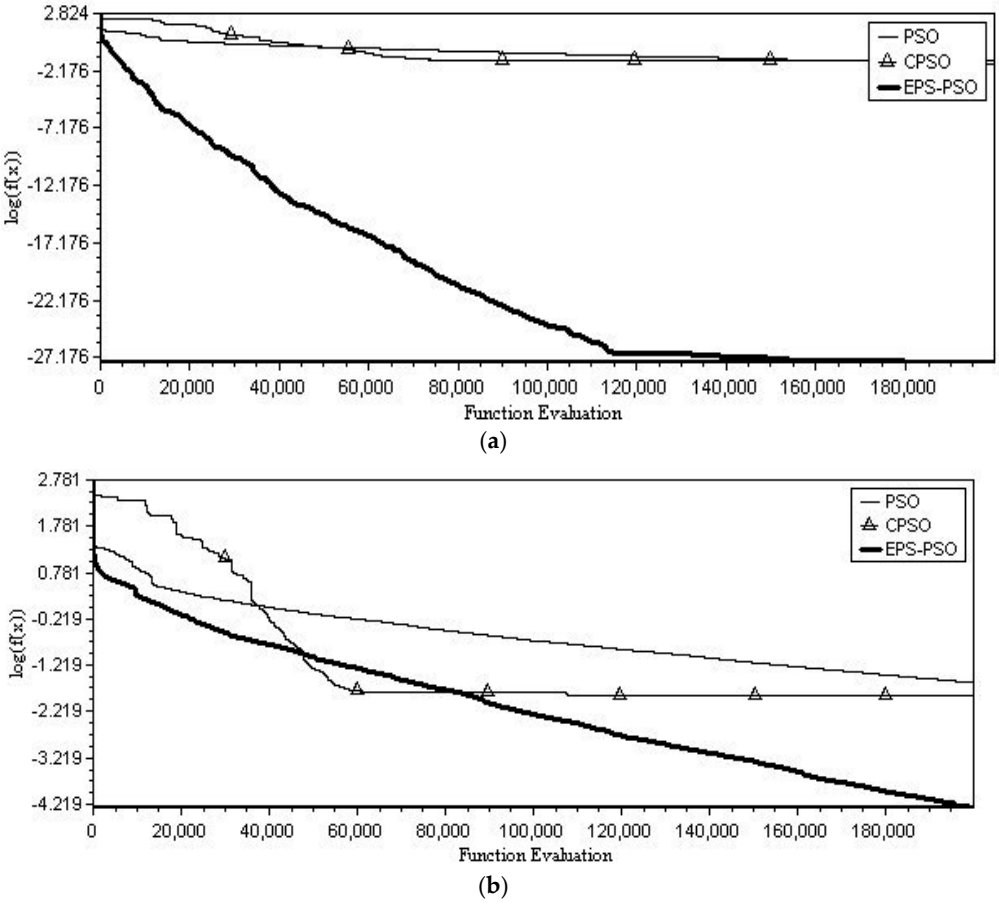


Figure 3. The best convergence results versus function evaluation among 50 independent runs for Rosenbrock (f_1): (**a**) unrotated case, (**b**) rotated case.

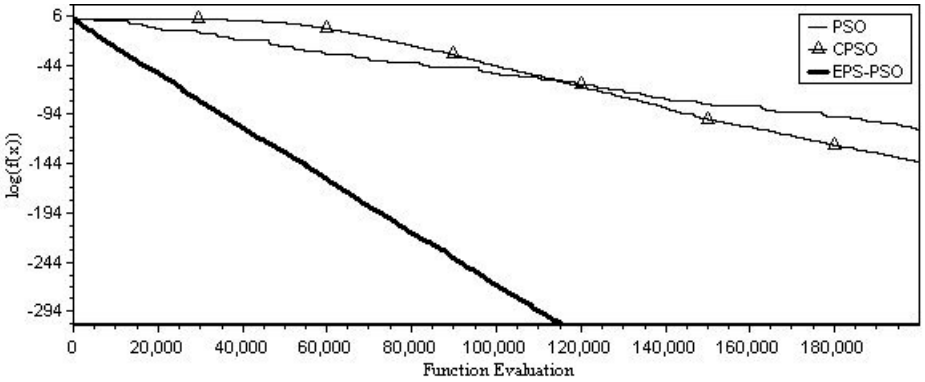
The computational results of the Quadric function solved by the 6 algorithms are tabulated in Table 3. All the algorithms perform well for the unrotated case; the EPS-PSO algorithm converges perfectly to the optimal point without any function error. Although the Quadric function is convex and unimodal, the coordinate rotation really makes it difficult to solve. The EPS-PSO algorithm is only slightly influenced by the coordinate rotation and still yields competitive performance. The other 5 algorithms cannot provide satisfactory performance. The best convergence results among 50 independent runs are displayed in Figure 4. For the unrotated case, the EPS-PSO algorithm converges the way faster than the other algorithms. In spite of the coordinate rotation as exhibited in Figure 4b, the EPS-PSO algorithm is still able to continuously improve the function error as iterations elapsed. Surprisingly, the traditional PSO and CPSO algorithms fail to solve the rotated Quadric function and the improvement in the function error becomes stagnated after 6×10^4 function evaluations.

The Ackley's Function is a multimodal function with many local minima positioned on a regular grid. The computational results are shown in Table 4. For the un-rotated case, the traditional PSO algorithm cannot solve the problem successfully; the 4 CPSO algorithms perform quite well except the CPSO$-S_6$ algorithm. The performance of the EPS-PSO algorithm deteriorates quickly as the re-initialization period increases. The global optimum is attained as $t = 100$. The explanation of how the EPS-PSO algorithm works using $t = 100$, 500 is that re-initializing the co-search swarm more often definitely helps the search jump out of the local optima. For the rotated case, all the 6 algorithms are seriously affected by the coordinate rotation except for the CPSO$-H_6$ algorithm with $s = 20$ and the EPS-PSO algorithm with $t = 500$. As before, the performance of the EPS-PSO algorithm becomes worse as t increases. The best convergence results are plotted in Figure 5. For the un-rotated case, the EPS-PSO algorithm converges to the global optimum after 4×10^4 function evaluations. For the

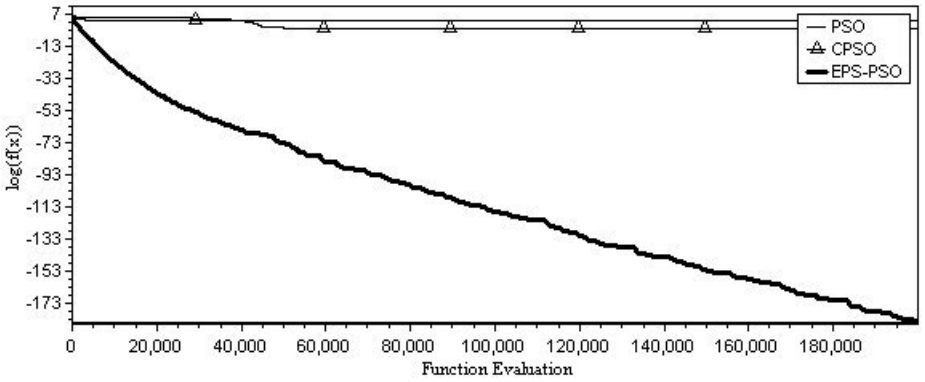
rotated case, the traditional PSO algorithm is trapped in a local solution, as can be seen from Figure 5b; the CPSO and EPS-PSO algorithms perform equally well.

Table 3. Computational results of Quadric obtained by using 6 PSO-based algorithms after 2×10^5 function evaluations, averaged over 50 independent runs.

Algorithm	s	t	Mean(Unrotated)	Mean(Rotated)
PSO	10	—	2.11e+00 ± 6.11e+00	6.11e+02 ± 3.07+e02
	15	—	3.73e–71 ± 2.72e–71	7.15e+02 ± 1.73+e02
	20	—	3.32e–95 ± 1.59e–96	3.82e+02 ± 6.12+e01
CPSO-*S*	10	—	1.72e–126 ± 7.18e–126	4.53e+02 ± 4.21+e02
	15	—	2.72e–90 ± 2.47e–89	6.81e+03 ± 2.96+e03
	20	—	1.17e–67 ± 8.87e–66	2.12e+03 ± 5.81+e03
CPSO-*H*	10	—	1.26e–93 ± 1.93e–92	1.41e+01 ± 5.59+e01
	15	—	2.81e–80 ± 1.02e–79	3.75e+02 ± 3.11+e02
	20	—	8.31e–61 ± 3.18e–62	2.62e+02 ± 2.31+e02
CPSO-S_6	10	—	1.61e–07 ± 7.93e–07	2.05e+03 ± 3.18+e03
	15	—	2.12e–05 ± 4.06e–05	1.62e+03 ± 4.83+e03
	20	—	8.22e–05 ± 4.75e–05	2.11e+03 ± 2.12+e03
CPSO-H_6	10	—	4.13e–63 ± 2.11e–63	1.63e+03 ± 8.61+e03
	15	—	7.12e–46 ± 1.69e–45	1.42e+02 ± 1.17+e02
	20	—	6.72e–27 ± 1.12e–28	8.64e+03 ± 4.92+e03
EPS-PSO	20	100	0.00e+00 ± 0.00e+00	1.18e–120 ± 1.90e–121
	20	500	0.00e+00 ± 0.00e+00	2.11e–140 ± 4.18e–140
	20	1000	0.00e+00 ± 0.00e+00	8.35e–183 ± 6.29e–184
	20	5000	0.00e+00 ± 0.00e+00	6.97e–157 ± 8.06e–157
	20	10000	0.00e+00 ± 0.00e+00	1.09e–112 ± 3.18e–112



(a)



(b)

Figure 4. The best convergence results versus function evaluation among 50 independent runs for Quadric (f_2): (**a**) unrotated case, (**b**) rotated case.

Table 4. Computational results of Ackley obtained by using 6 PSO-based algorithms after 2×10^5 function evaluations, averaged over 50 independent runs.

Algorithm	s	t	Mean(Unrotated)	Mean(Rotated)
PSO	10	—	6.13e+00 ± 8.01e+00	7.32e+00 ± 2.52e+00
	15	—	4.62e+00 ± 6.11e+00	9.65e+00 ± 1.88e+00
	20	—	2.86e+00 ± 4.72e+00	8.12e–07 ± 1.70e–07
CPSO-*S*	10	—	4.12e–14 ± 7.25e–14	1.22e+01 ± 1.72e+00
	15	—	6.18e–14 ± 6.79e–14	2.11e+01 ± 8.92e+00
	20	—	3.72e–14 ± 8.24e–14	8.16e–01 ± 4.37e–01
CPSO-*H*	10	—	1.63e–14 ± 2.92e–15	5.23e+01 ± 8.70e+00
	15	—	7.22e–14 ± 5.12e–15	4.13e+00 ± 2.18e+00
	20	—	1.71e–14 ± 4.66e–15	3.16e+01 ± 1.92e+00
CPSO-S_6	10	—	8.12e–07 ± 1.70e–07	3.08e+01 ± 1.06e+00
	15	—	4.61e–05 ± 4.11e–05	9.21e+01 ± 7.43e+00
	20	—	7.13e–05 ± 4.16e–05	5.25e+00 ± 5.26e+00
CPSO-H_6	10	—	3.84e–11 ± 6.82e–11	8.12e+00 ± 4.04e+00
	15	—	1.15e–12 ± 2.63e–12	6.02e–04 ± 6.51e–04
	20	—	1.72e–12 ± 1.42e–12	2.11e–11 ± 1.23e–11
EPS-PSO	20	100	0.00e+00 ± 0.00e+00	2.44e+00 ± 1.84e+00
	20	500	6.51e–19 ± 6.51e–19	2.26e–13 ± 1.64e–12
	20	1000	3.72e–03 ± 2.95e–03	1.93e–07 ± 1.53e–08
	20	5000	3.31e–02 ± 3.06e–02	1.17e+00 ± 1.94e+00
	20	10000	2.05e+00 ± 2.50e+00	1.85e+00 ± 1.66e+00

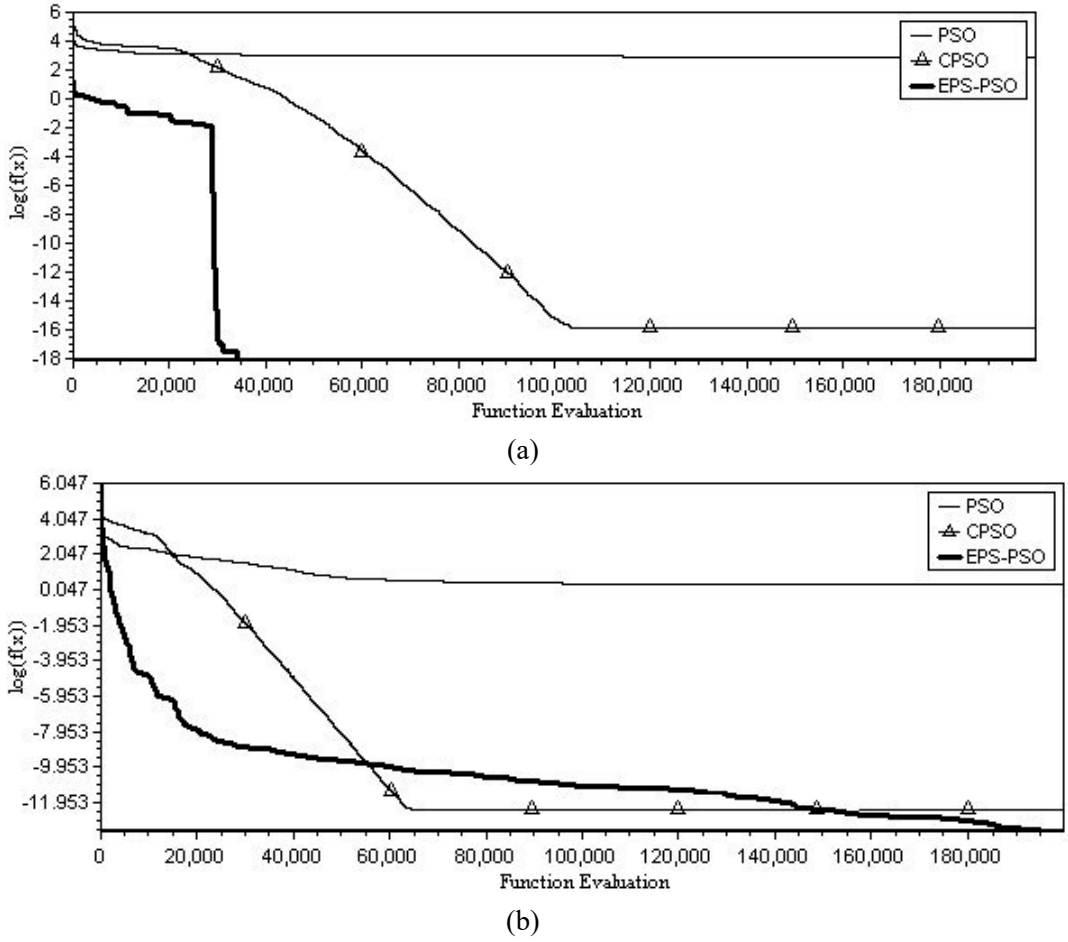


Figure 5. The best convergence results versus function evaluation among 50 independent runs for Ackley (f_3): (**a**) unrotated case, (**b**) rotated case.

The Rastrigin function is a typical nonlinear multimodal function, which contains only one global optimum with a large number of local minima, making the problem more difficult to solve. The computational results are listed in Table 5. For the un-rotated case, only the CPSO−S algorithm, the CPSO−H algorithm and the EPS-PSO algorithm with $t = 500,\ 1000$ can locate the global optimum. Once the search space is rotated, only the EPS-PSO algorithm with $t = 500$ can produce satisfactory performance. The best convergence plots are shown in Figure 6.

Table 5. Computational results of Rastrigin obtained by using 6 PSO-based algorithms after 2×10^5 function evaluations, averaged over 50 independent runs.

Algorithm	s	t	Mean(Unrotated)	Mean(Rotated)
PSO	10	—	6.37e+01 ± 5.03e+00	6.28e+02 ± 5.38e+01
	15	—	7.48e+01 ± 3.06e+00	5.28e+02 ± 1.29e+01
	20	—	2.88e+01 ± 1.41e+00	5.49e+02 ± 7.16e+01
CPSO-S	10	—	0.00e+00 ± 0.00e+00	4.21e+01 ± 8.46e+01
	15	—	0.00e+00 ± 0.00e+00	7.73e+01 ± 4.43e+01
	20	—	0.00e+00 ± 0.00e+00	7.16e+01 ± 2.73e+01
CPSO-H	10	—	0.00e+00 ± 0.00e+00	7.08e+01 ± 6.83e+01
	15	—	0.00e+00 ± 0.00e+00	8.35e+01 ± 6.86e+01
	20	—	0.00e+00 ± 0.00e+00	8.06e+01 ± 9.92e+01
CPSO-S_6	10	—	3.29e–02 ± 7.73e–02	5.11e+01 ± 5.28e+01
	15	—	6.80e–02 ± 6.62e–02	4.16e+01 ± 4.34e+01
	20	—	6.59e–01 ± 8.03e–01	5.14e+01 ± 6.15e+01
CPSO-H_6	10	—	9.45e–01 ± 2.12e–01	9.32e+01 ± 3.72e+01
	15	—	4.37e–01 ± 2.15e–01	4.12e+01 ± 5.84e+01
	20	—	6.08e–01 ± 5.34e–01	5.20e+01 ± 7.46e+01
EPS-PSO	20	100	5.81e–13 ± 6.67e–13	1.29e–04 ± 1.21e–05
	20	500	0.00e+00 ± 0.00e+00	3.30e–11 ± 8.32e–12
	20	1000	0.00e+00 ± 0.00e+00	6.17e–02 ± 1.34e–02
	20	5000	4.08e–03 ± 1.40e–02	1.05e–02 ± 8.28e–01
	20	10000	8.89e–02 ± 9.31e–02	7.57e–01 ± 5.41e–01

Table 6. Computational results of Griewank obtained by using 6 PSO-based algorithms after 2×10^5 function evaluations, averaged over 50 independent runs.

Algorithm	s	t	Mean(Unrotated)	Mean(Rotated)
PSO	10	—	1.05e+01 ± 4.36e+02	5.18e+01 ± 2.18e+01
	15	—	7.42e+01 ± 8.78e+02	1.48e+01 ± 2.43e+01
	20	—	6.12e+01 ± 6.40e+02	2.76e+01 ± 2.15e+01
CPSO-S	10	—	2.16e–02 ± 2.77e–02	5.25e–01 ± 9.60e–01
	15	—	5.12e–02 ± 6.05e–02	6.78e–01 ± 6.45e–01
	20	—	9.43e–03 ± 7.13e–03	6.21e–01 ± 5.86e–01
CPSO-H	10	—	2.88e–02 ± 3.04e–02	5.10e–01 ± 3.44e–01
	15	—	2.18e–02 ± 4.28e–02	3.15e–01 ± 9.01e–01
	20	—	2.74e–02 ± 1.86e–02	4.81e–01 ± 2.78e–01
CPSO-S_6	10	—	4.19e–02 ± 6.72e–02	5.12e–01 ± 2.48e–01
	15	—	4.48e–02 ± 5.49e–02	7.14e–01 ± 1.46e–01
	20	—	4.18e–02 ± 7.29e–02	1.06e–01 ± 2.11e–01
CPSO-H_6	10	—	6.98e–02 ± 5.40e–02	1.40e–01 ± 1.62e–01
	15	—	7.16e–02 ± 1.13e–02	1.80e–01 ± 4.05e–01
	20	—	5.42e–02 ± 5.14e–02	4.21e–01 ± 1.94e–01
EPS-PSO	20	100	2.20e–05 ± 3.32e–05	3.62e–04 ± 7.40e–05
	20	500	2.09e–08 ± 6.68e–08	7.28e–06 ± 1.01e–07
	20	1000	4.01e–09 ± 3.95e–09	5.51e–06 ± 1.95e–06
	20	5000	2.04e–09 ± 2.11e–09	8.11e–05 ± 9.04e–05
	20	10000	7.44e–03 ± 3.08e–03	5.02e–05 ± 7.09e–05

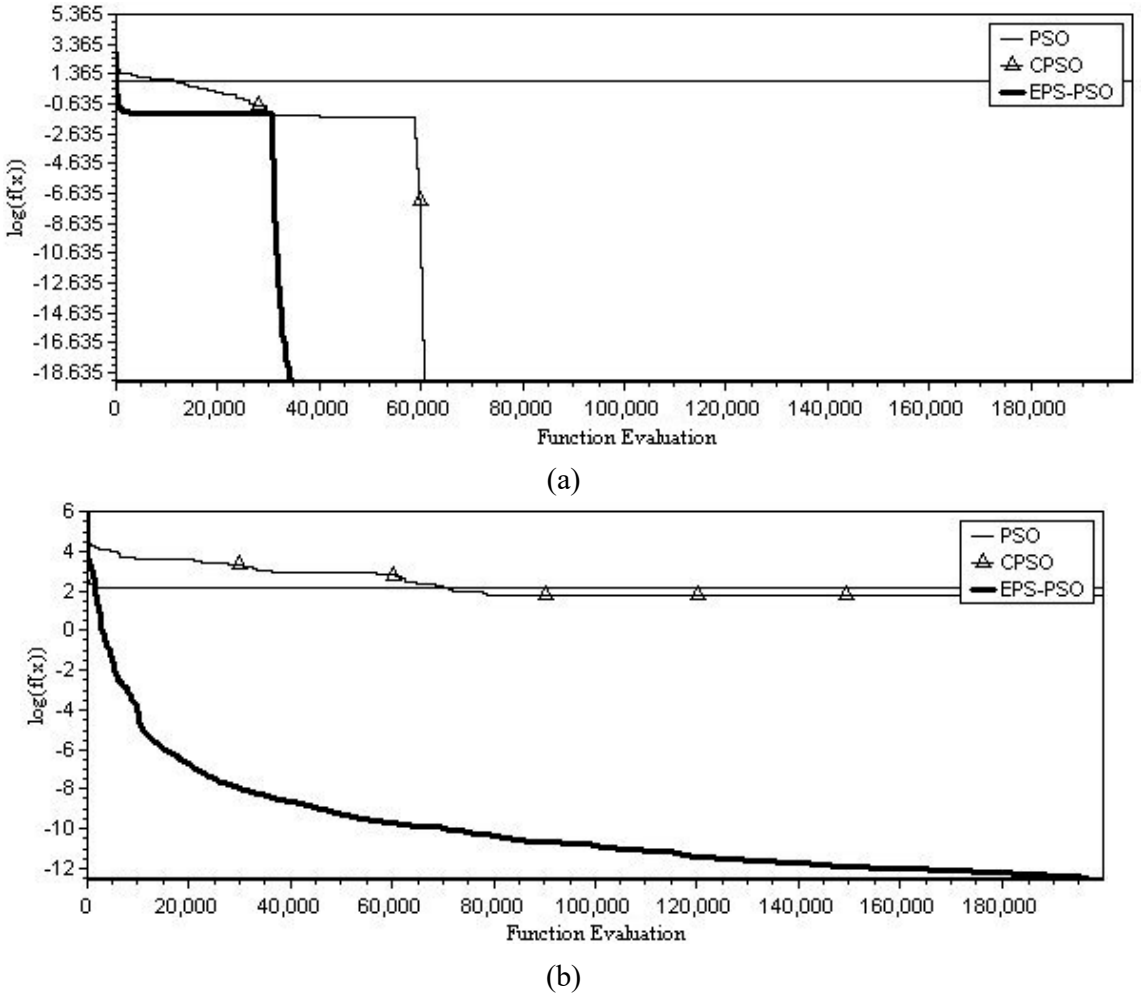


Figure 6. The best convergence results versus function evaluation among 50 independent runs for Rastrigin (f_4): (**a**) unrotated case, (**b**) rotated case.

Table 6 shows that the EPS-PSO algorithm performs better than the other PSO algorithms in all experiments on the un-rotated Griewank function. For the rotated case, the EPS-PSO algorithm still performs best among the studied algorithms. The best convergence performance is plotted in Figure 7. It can clearly be seen from the figure that the traditional PSO and CPSO algorithms have almost no improvement in the function error after 4×10^4 function evaluations due to the multimodal structure. The EPS-PSO algorithm converges really fast from the outset, but the convergence slows down after 4×10^4 function evaluations.

In Table 7, the performances between the proposed EPS-PSO and different PSO algorithms are compared for verifying the effectiveness of EPS-PSO. In the table, the t-test was used to test whether the proposed EPS-PSO algorithm can outperform significantly the other algorithms in the minimization case. In terms of the two-sample t-test, the null and alternative hypotheses on the difference of the objective functions achieved are constructed to be $H_0: \mu_{EPS-PSO} - \mu_{other\ PSO} \geq 0$ and $H_1: \mu_{EPS-PSO} - \mu_{other\ PSO} < 0$, respectively. The p-value of the test statistic less than 0.05 indicates that the proposed EPS-PSO algorithm generates a significantly smaller objective function than the compared PSO variant, which is marked "○". Conversely, the p-value greater than 0.05 indicates that the EPS-PSO algorithm does not return a significantly smaller objective function than the compared PSO variant, which is marked as "×". Overall, the EPS-PSO algorithm exhibits a significantly better performance than the other five algorithms except the functions Ackley and Rastrigin in the unrotated case. For the rotated case, the EPS-PSO algorithm outperforms overwhelmingly the other 5 PSO variants.

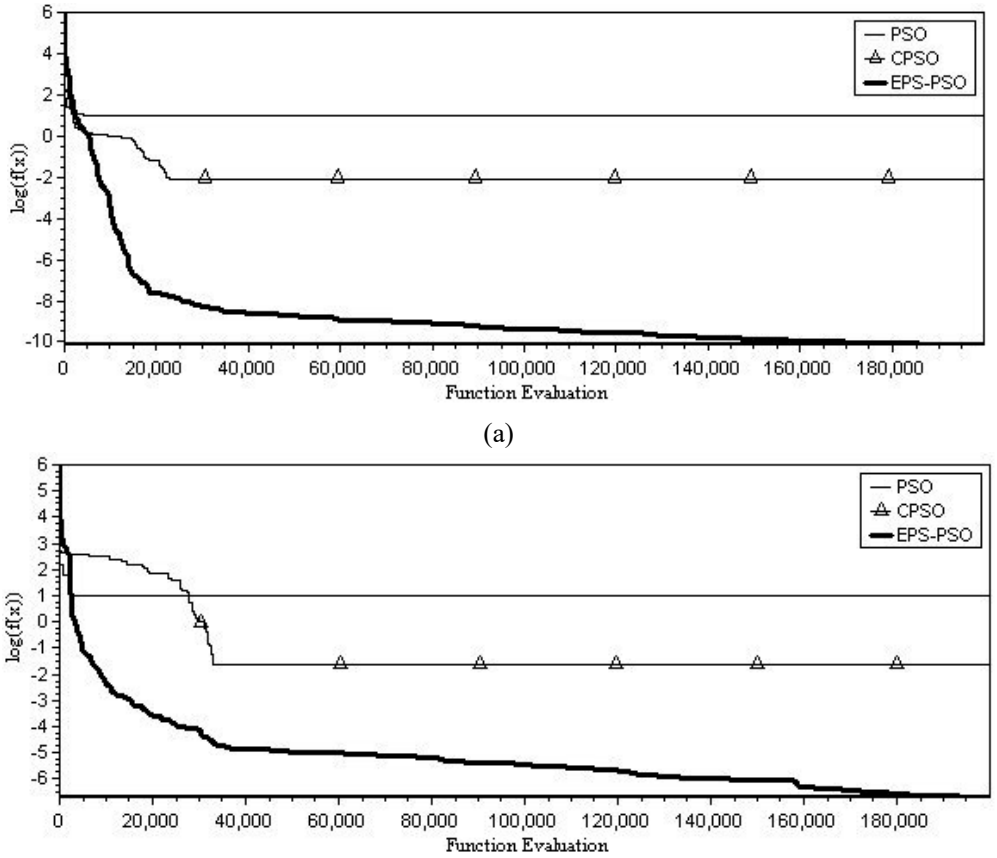


Figure 7. The best convergence results versus function evaluation among 50 independent runs for Griewank (f_5): (**a**) unrotated case, (**b**) rotated case.

Table 7. The performance comparisons between the proposed EPS-PSO algorithm and five different PSO variants over five benchmark functions.

Functions		PSO	CPSO-S	CPSO-H	CPSO-S_6	CPSO-H_6
Rosenbrock	EPS-PSO Unrotated	○	○	○	○	○
	EPS-PSO Rotated	○	○	○	○	○
Quadratic	EPS-PSO Unrotated	○	○	○	○	○
	EPS-PSO Rotated	○	○	○	○	○
Ackley	EPS-PSO Unrotated	○	×	×	×	×
	EPS-PSO Rotated	×	○	○	○	×
Rastrigin	EPS-PSO Unrotated	○	×	×	○	○
	EPS-PSO Rotated	○	○	○	○	○
Griewank	EPS-PSO Unrotated	○	○	○	○	○
	EPS-PSO Rotated	○	○	○	○	○

As evidenced by the foregoing computational results, the enhanced partial search (EPS) mechanism has proved a valuable investment that helps to prevent the particles from being trapped in the local optima more likely than purely using the traditional PSO search. The re-initialization period t is designed to control the frequency of how the co-search swarm is reinitialized. The five levels of t are tested for the five unrotated and rotated benchmark functions, and the EPS-PSO algorithm using the level of 500 iterations produces the most reliable performance. To summarize, the better results obtained by the proposed EPS-PSO algorithm are attributed largely to (i) the topology of two sub-swarms between which the communication is carried out periodically and (ii) the re-initialization of the co-search swarm that is augmented to diversify local exploitation.

6. Conclusions and Directions of Future Research

This paper presents a cooperative multi-swarm structure in order to enhance the searching ability of the traditional PSO algorithm. The new search mechanism is called the enhanced partial search (EPS), making the new algorithm dubbed EPS-PSO. The EPS is a rerandomization strategy that works on an auxiliary swarm, termed the co-search swarm. The intension is to mobilize the particles and then make the swarm have more chances to explore different search areas. Moreover, the EPS mechanism has been implemented to allow the co-search swarm to share information with the traditional swarm during the evaluation process. Five benchmark functions in the unrotated and rotated versions are used as the test bed to compare the EPS-PSO algorithm to the traditional PSO and 4 different cooperative PSO algorithms. Five levels of the re-initialization period are examined, $t = 100,\ 500,\ 1000,\ 5000,\ 10,000$. The EPS-PSO algorithm performs best when the re-initialization period is set to 500 iterations. The comparison results show that the proposed algorithm improves upon the traditional PSO algorithm and outperforms the other 4 cooperative PSO algorithms. In other words, the EPS is shown to be a simple but effective way that helps to prevent the swarm from becoming stuck in the local optima.

Built upon the current research, there are still several PSO-related research topics that deserve further scrutiny. The first potential investigation is to extend the propose EPS-PSO algorithm to constrained optimization. In this case, a mechanism to handle the constraint violation should be invented and comprehensively tested. A further study can be done by comparing the proposed EPS-PSO algorithm to some PSO variants equipped with the swarm collapse detection mechanism. Furthermore, the co-search swarm strategy may also be useful for PSO dealing with noisy function evaluations.

Author Contributions: This is a joint work of the two authors; nevertheless, each author was especially in charge of his expert and capability: S.K.S.F. for conceptualization, methodology, investigation and formal analysis, S.K.S.F. and C.H.J. for validation, original draft preparation and writing.

Funding: This research was partially funded by Ministry of Science and Technology, grant numbers MOST 105-2221-E-027-052-MY3 and MOST 105-2221-E-027-071-MY3.

Acknowledgments: The authors are deeply grateful to J.M. Chang's great efforts for conducting an earlier experimental study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley Publishing Inc.: Reading, MA, USA, 1989.
2. Kennedy, J.; Eberhart, R.C. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
3. Dorigo, M.; Caro, G.D. Ant colony optimization: A new meta-heuristic. In Proceedings of the Congress on Evolutionary Computation, Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477.
4. Shi, Y.; Eberhart, R.C. A modified particle swarm optimizer. In Proceedings of the IEEE International Conference on Evolutionary Computation, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.

5. Wang, X.-H.; Jun-Jun, L. Hybrid particle swarm optimization with simulated annealing. In Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, China, 26–29 August 2004; Volume 4, pp. 2402–2405.
6. Cui, Z.H.; Zeng, J.C.; Cai, X.J. A new stochastic particle swarm optimizer. In Proceedings of the 2004 Congress on Evolutionary Computation, Piscataway, NJ, USA, 19–23 June 2004; pp. 316–319.
7. Zahara, E.; Fan, S.K.-S.; Tsai, D.M. Optimal multi-thresholding using a hybrid optimization approach. *Patten Recognit. Lett.* **2004**, *26*, 1082–1095. [CrossRef]
8. Fan, S.-K.S.; Liang, Y.C.; Zahara, E. A Hybrid Simplex Search and Particle Swarm Optimization for the Global Optimization of Multimodal Functions. *Eng. Optim.* **2004**, *36*, 401–418. [CrossRef]
9. Fan, S.-K.S.; Zahara, E. A hybrid simplex search and particle swarm optimization for unconstrained optimization. *Eur. J. Oper. Res.* **2007**, *181*, 527–548. [CrossRef]
10. Van den Bergh, F.; Engelbrecht, A.P. A Cooperative Approach to Particle Swarm Optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 225–239. [CrossRef]
11. Angeline, P.J. Using selection to improve particle swarm optimization. In Proceedings of the IEEE International Conference on Evolutionary Computation, Anchorage, AL, USA, 4–9 May 1998; pp. 84–89.
12. Lovbjerg, M.; Rasmussen, T.K.; Krink, T. Hybrid particle swarm optimizer with breeding and subpopulations. In Proceedings of the Third Genetic and Evolutionary Computation Congress, San Diego, CA, USA, 7–11 July 2001; pp. 469–476.
13. Lovbjerg, M.; Krink, T. Extending particle swarm optimizers with self-organized criticality. In Proceedings of the Fourth Congress on Evolutionary Conference, Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1588–1593.
14. Rada-Vilela, J.; Johnston, M.; Zhang, M. Population statistics for particle swarm optimization: Resampling methods in noisy optimization problems. *Swarm Evol. Comput.* **2014**, *17*, 37–59. [CrossRef]
15. Taghiyeh, S.; Xu, J. A new particle swarm optimization algorithm for noisy optimization problems. *Swarm Intell.* **2016**, *10*, 161–192. [CrossRef]
16. Eberhart, R.C.; Dobbins, R.C.; Simpson, P. *Computational Intelligence PC Tools*; Academic Press Professional: Boston, MA, USA, 1996.
17. Potter, M.A.; de Jong, K.A. *A Cooperative Coevolutionary Approach to Function Optimization the Third Parallel Problem Solving from Nature*; Springer: Berlin, Germany, 1994; pp. 249–257.
18. Van den Bergh, F.; Engelbrecht, A.P. Effects of swarm size on cooperative particle swarm optimizers. In Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, USA, 7–11 July 2001; pp. 892–899.
19. Shang, Y.W.; Qiu, Y.H. A note on the extended Rosenbrock function. *Evol. Comput.* **2006**, *14*, 119–126. [CrossRef] [PubMed]
20. Črepinšek, M.; Liu, S.H.; Mernik, M. Replication and comparison of computational experiments in applied evolutionary computing: Common pitfalls and guidelines to avoid them. *Appl. Soft Comput.* **2014**, *19*, 161–170. [CrossRef]
21. Shi, Y.; Eberhart, R.C. Empirical study of particle swarm optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, NJ, USA, 6–9 July 1999; pp. 1945–1950.
22. Salomon, R. Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions. *BioSystems* **1996**, *39*, 263–278.