Applied Mathematics & Information Sciences
An International Journal

A Modified PSO Algorithm for Numerical Optimization Problems

Hsin-Chuan Kuo, Jeun-Len Wu and Ching-Hai Lin

Department of Systems Engineering and Naval Architecture, National Taiwan Ocean University, Keelung City, 202, Taiwan

Received: 7 Sep. 2012, Revised: 21 Nov. 2012, Accepted: 23 Dec. 2012
Published online: 1 May 2013

Abstract: By successively employing the interval search method, we developed the proposed algorithm MPSO, introducing three creative position vectors to replace the three worst fitness particles among the population in the PSO, to overcome the premature convergence situation that occurs when a problem with a large number of variables and (or) multiple optima is solved.
The results obtained by applying the MPSO and the PSO on 6 benchmark functions show that, except for the randomly shifted Rosenbrock functions, the MPSO can successfully secure a solution that is close to the exact solution for each of the remaining five functions. We also showed that all benchmark functions are solvable by the MPSO if the maximum number of generations is raised to be as high as possible. With regard to the PSO′s performance for the three different numbers of variables, it fails to obtain a solution that is close to the exact solution for all of the tested functions except for the Sphere function with 30 variables.

Keywords: Particle Swarm Optimization, The Interval Search method, Constrained Optimization Problems, Global optimization.

1. Introduction

The population-based evolutionary computation technique of Particle Swarm Optimization (PSO) was originally developed by Kennedy and Eberhart [1], inspired by animal social behaviors such as bird flocking and fish schooling. When a bird in a flock tries to find food, it uses its own knowledge and experience as well as its neighbors experience. Hence, the birds in a flock are analogous to particles in the context of PSO - particles that are initially selected randomly within the search space of a problem of interest. In PSO, the motion of each particle is determined mainly by its velocity, which is updated for each generation according to three main features: habit, self-cognition and sociality.

As a simple and easily implemented mathematical model, the PSO algorithm is applicable for a wide range. However, the strong cohesive traction amid the particles in PSOs always cause them to aggregate quickly, and hence, their ability to explore outside of the aggregation is reduced. This feature often results in a final solution which converges to a local optimum rather than a global optimum when multi-modal problems are solved. Moreover, the PSO gives favorable results only for a problem with a small number of variables; for a problem with a large number of variables, it has difficulty in obtaining a solution that is close to the problems exact solution. Thus, there are many variants of the PSO that have been developed to improve its performance by modifying its searching strategy or by tuning the relevant parameters and some of these variants are described briefly below.

Suganthan [2], in 1999, and Ratnaweera *et al.* [3], in 2004, discussed learning factors, linearly decreasing inertia weights and the constriction factor of generational adjustment to improve a global search; Chatterjee and Siarry [4], in 2004, proposed a non-linear dynamic inertia weight. Liu *et al.* [5], in 2005, suggested a chaotic system that modifies the inertia weight according to the relationship between every particle and the average fitness value. Bin *et al.* [6], in 2006, proposed using a random number between 0 and 1 to increase the inertia weight disturbance values. Yang *et al.* [7], in 2007, proposed a dynamic adjustment scheme for the inertia weight, assigning every particle its own inertia weight. Xiang *et al.* [8], in 2006, proposed a time delay to control the diversification of the particle swarm and, thereby, to maintain the swarm activity. In 2006, Fan and Zahara [9] proposed a hybrid of the simplex method and the PSO, using the individual moving mode l of the simplex method to replace the particle moving model of the PSO.

* Corresponding author e-mail: khc@ntou.edu.tw

Yeh [10] incorporated a local search method into the PSO algorithm to construct a hybrid PSO, in which the parallel population-based evolutionary ability of the PSO and the local searching behavior are reasonably combined.

The above PSO performance-enhancing strategies can be categorized in three different ways: adjusting both the inertia weight and the learning factors, combining the PSO with other numerical algorithms to form a hybrid PSO, and introducing a local search technique to increase the particles searching performance. The variants of the PSO from the three strategies have improved the performance of the original PSO, but unfortunately, a limited level of improvement has been reached so far.

It is believed that the relationship among the variables of a problem of interest, separable or non-separable, would definitely challenge the performance of an optimization solver.

If a separable problem is considered, then a successive interval search method is suggested and is directly applied in search of the problems solution; this approach would obtain a solution that is closest to the exact solution. The successive interval method is derived in the study from an interval search method [11], which is only for a one-variable optimization problem. In implementing the successive interval search method, the solution of each variable can be screened by fixing the remaining variables with given values by successively narrowing the search domain of the variable until the size of the latest domain satisfies the criteria set. The details of how to extend the interval search method into the successive interval search method is to be described in the next paragraph.

The proposed algorithm is similar to one using the above-mentioned second strategies for enhancing the PSOs performance. The successive interval search method in the proposed algorithm is implemented after numerous PSO generations to generate three position vectors that replace the worst three fitness particles in the population. The new three particles then act as stimulant agents that provide traction to pull the other particles in the population out of the stagnant situation for exploration in the direction of the problems exact solution. Depending on a problems characteristics or the number of variables in a problem of interest, in the evolutionary process of searching the solution closest to the problems exact solution by the proposed algorithm, the successive interval method could be implemented a number of times.

To verify the proposed algorithm, we selected 6 benchmark functions with dependent or independent variables, for which six of the functions possess a single optimum, and the remaining five have multiple optima. Additionally, to understand the performance of the proposed algorithm in relation to the increasing numbers of variables, three different numbers of variables were considered (30, 50 and 100).

2. Related measures that were used when developing the proposed algorithm

2.1. Definition of Separable and Non-separable functions

A function possessing variables that are independent of one another is called a separable function; otherwise, it is a non-separable function. This distinction, in the mathematical terms given below, was recently defined by Yang *et al.* [12].

Definition : $f(\mathbf{x})$ is called a separable function if $\forall k \in \{1,m\}$ and

$$\left.\begin{array}{l}\mathbf{x}\in\Re,\ \mathbf{x}=\{x_1,\cdots,x_k,\cdots,x_m\}\\ \mathbf{x}'\in\Re,\ \mathbf{x}=\{x_1,\cdots,x'_k,\cdots,x_m\}\end{array}\right\}\Rightarrow f(\mathbf{x})\le f(\mathbf{x}')$$

Imply

$$\left.\begin{array}{l}\mathbf{y}\in\Re,\ \mathbf{y}=\{y_1,\cdots,y_k,\cdots,y_m\}\\ \mathbf{y}'\in\Re,\ \mathbf{y}=\{y_1,\cdots,y'_k,\cdots,y_m\}\end{array}\right\}\Rightarrow f(\mathbf{y})\le f(\mathbf{y}')$$

Otherwise, $f(\mathbf{x})$ is called a non-separable function.

In the application of evolutionary algorithms, the interaction among the variables of a non-separable function has been considered to be the main culprit in incurring the curse of dimensionality, which implies that the performance of the algorithms deteriorates rapidly as the dimensionality of the search space increases.

Therefore, if one can distinguish a problem as being separable or non-separable, then this distinction will provide helpful information on what computational algorithm should be employed for the problems solution to be effective and efficient.

2.2. Successive Interval Search Method (SIS)

The interval search method described above is then extended, in this study, to a problem with n variables, in other words, $f(x_1,\ldots,x_n)$ and $x_i \in [a_i,b_i]$, and the extended search mechanism is called the successive interval search method, abbreviated as SIS.

In the SIS, we define the number of sub-intervals that are divided at each search stage for each variable denoted by n_{di}. The variables $x_1,\ldots,x_n$ are initially randomly selected, corresponding to their respective specified domains.

Next, we apply the interval search method on the first variable x_1 by fixing the remaining variables with given values. Then, the variable vector of the problem changes with changes in its first component x_1. Therefore, similar to the process of the interval search method for the function with a single variable, we can gradually zero in on the best value of x_1, denoted as x_{1b}, with which the variable vector obtained will give the minimum function value. Next, we proceed with the same procedure for the

second variable x_2 to determine the value of x_{2b}. Similarly, we keep implementing the interval search method on one variable at a time, and we can determine $x_{3b}, x_{4b}, \ldots, x_{nb}$.

If a problem of interest possesses variables that are non-interacting with each other, then the vector determined by the SIS method $\mathbf{x}_b = \{x_{1b}, x_{2b}, \ldots, x_{nb}\}$ is actually the problems optimum solution. Conversely, for a problem with variables that interact with each other, the solution by SIS $\mathbf{x}_b = \{x_{1b}, x_{2b}, \ldots, x_{nb}\}$ is no longer the optimal solution; however, we propose that the solution could provide information when the particles in the PSO are trapped. We then attempt to incorporate this strategy into the PSO algorithm for the proposed algorithm.

3. The Modified Particle Swarm Optimization (MPSO)

In optimizing a problem, we found that if it is possible to determine whether a problem is separable or non-separable prior to attempting to solve the problem, then one can determine a reliable solution more effectively and efficiently. For a separable problem, we suggest that the SIS is the best solver compared with other heuristic evolutionary algorithms. However, for a non-separable problem, the mutual dependence among the variables cripples the fundamental structures of the SIS, which is thus not an appropriate solver for the problem. For a non-separable problem, one usually resorts to employing a heuristic evolutionary algorithm.

Among all of the types of well-known algorithms, the PSO is a simple, fast and easily implemented evolution algorithm, but its performance usually provides an unreliable result when a complicated problem is solved, such as a multi-modal problem or a problem with a large number of variables. Therefore, many variants of PSOs have been developed to improve its performance on complicated problems by modifying the parameters defined in the PSO or by introducing local search methods or cooperating with other computational algorithms.

Because the SIS is well suited for separable problems, the possibility of combining the SIS with the PSO for problems that are either separable or non-separable is proposed here. Moreover, the mechanisms for applying the solution obtained by the SIS are given, and a helpful guidance strategy for the particles in the population of the PSO is also created.

Therefore, we have developed an algorithm, called the modified Particle Optimization (MPSO), which begins with distinguishing a problem type by figures and then follows the improvement of the PSO performance with the assistance of the SIS. The flowchart of the MPSO is shown in Fig. 1.

3.1. Three different position vectors

In the proposed algorithm, we introduce three creative positions, which are denoted by $\mathbf{x}_{avg}^{pbest}$, $\mathbf{x}_{SIS}$ and $\mathbf{x}_{Disturb}$, to replace the three poorest fitness particles among the population of particles in the PSO. This replacement aims to improve both the exploitation and the exploration of the particles, which stay in a stagnant situation.

3.2. Parameters set in the MPSO

When implementing the MPSO, the parameters used in the PSO must be specified for the weight coefficient ω, the learning coefficients c_1 and c_2, and the number of particles in the population N_p; in SIS, the replacement rate N_{rplc} and the number of points n_{di} to divide the search space for each variable must be used. Additionally, the convergence conditions, the maximum number of generations N_{max}, and the convergence criterion ε are included as well.

4. Experimental Results and Discussion

4.1. Benchmark functions

To verify the MPSO, we selected 6 man-made benchmark problems, which are displayed as follows. The domains and exact solutions of each functions are listed in Table 1. The last two functions are randomly shifted functions with $\mathbf{z} = \mathbf{x} - \mathbf{x}_o$, where the vector $\mathbf{x}_o$ is randomly created.

F1 : Sphere

$$F_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$$

F3 : Rosenbrock

$$F_3(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right]$$

F4 : Ackley

$$F_4(\mathbf{x}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$$

F7 : Schwefel

$$F_7(\mathbf{x}) = \sum_{i=1}^{n} x_i sin\left(\sqrt{|x_i|}\right)$$

F10 : Random Shifted Rosenbrock

$$F_{10}(\mathbf{z}) = \sum_{i=1}^{n-1} \left[100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2\right]$$

Table 1 Domains and exact solutions of benchmark functions.

Function	Domains $i = 1,2,\ldots,n$	Exact solution
F1	$-100 \le x_i \le 100$	$F_1(\mathbf{0}) = 0$
F3	$-30 \le x_i \le 30$	$F_3(\mathbf{1}) = 0$
F4	$-32 \le x_i \le 32$	$F_4(\mathbf{0}) = 0$
F7	$-500 \le x_i \le 500$	$F_7(-\mathbf{420.97}) = -418.98 \times n$
F10	$-30 \le x_i \le 30$	$F_{10}(\mathbf{z}) = 0$
F11	$-32 \le x_i \le 32$	$F_{11}(\mathbf{z}) = 0$

F11 : Random Shifted Ackley

$$F_{11}(\mathbf{z}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi z_i)\right) + 20 + e$$

4.2. In Comparison with the original PSO

In the computation, the relevant parameters were given values as follows: the number of particles in the population N_p=80; the weight coefficient $\omega = 0.6$; the learning coefficients $c_1 = 1.5$ and $c_2 = 2.0$; the number of subintervals n_{di}=9 ; the replacement rate N_{rplc}=250; the maximum number of generations N_{max}=50,000; and the convergence criteria $\varepsilon = 10^{-6}$. There are three different numbers of variables that were used for the comparison: $n = 30$, $n = 50$, and $n = 100$.

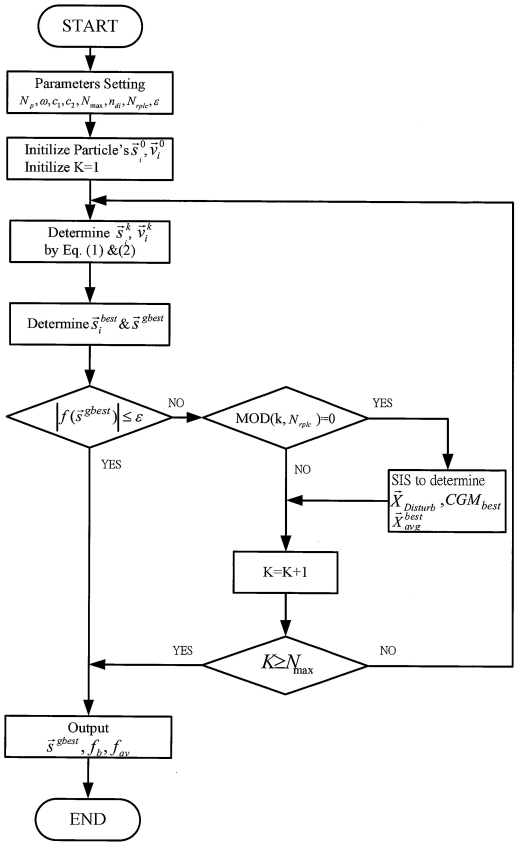

Figure 1 The flowchart for implementing MPSO.

4.3. Effect of the number of variables

Because the proposed algorithm MPSO is developed for the purpose of improving the PSO algorithm, the results determined by MPSO are compared with those of the PSO. To avoid a bias in the results, we performed each function by both of the algorithms, through 30 independent runs. To present the performance of both algorithms, we define a parameter f_{avg} , which averages the final solutions of the 30 runs, and a parameter f_b , which represents the best value among the 30 finished function values. Obviously, the less difference between f_b and f_{avg}, the more reliable the algorithm is. Moreover, we define the success rate as the rate of the number of final solutions that met the convergence criterion ε over the total of 30 runs.

Therefore, the optimization results obtained by both MPSO and PSO for all of the 6 functions with 30 variables, 50 variables and 100 variables are presented, respectively, in Table 2, Table 3, and Table 4.

For the optimization results of the functions with 30 variables, shown in Table 2, we found that the MPSO can determine the solution definitely (a success rate of 100%) and is close to the exact solution for each function except for the function F5, for which the MPSO determined a solution close to the exact solution with a success rate of 60%, meaning that the number of solutions close to the exact solution takes only 18 times among 30 runs. Concerning the results optimized by the PSO, only one function, F1, among the 6 tested functions satisfies the goal of a success rate of 100 %. Moreover, the PSO gives a success rate with a value of less than 50% for the functions F2 and F3. Additionally, the results for the last five functions, F4 to F6, indicate that the PSO cannot address solving these functions at all.

Next, the optimization results of the tested functions that had 50 variables are shown in Table 3. The results present almost the same performance for both MPSO and PSO as they did for 30 variables, except that the success rate determined by MPSO for the function F5 to 36.66% from 60%. Undoubtedly, the number of variables increases the difficulty of screening the solution by the MPSO. However, if we remove the limitation of the

Table 2 The results for the 6 tested functions, each with 30 variables.

	PSO		
function	f_{av}	f_b	Sucess Rate
F1	8.37E-07	5.89E-07	100%
F3	1.82E+04	1.00E-06	3.3%
F4	1.30E+01	8.46E-07	23.3%
F7	-6.71E+03	-8.11E+03	0
F10	4.31E+07	5.03E+06	0
F11	1.56E+01	2.03E+00	0
	MPSO		
function	f_{av}	f_b	Sucess Rate
F1	8.76E-07	6.81E-07	100%
F3	1.20E-06	9.88E-07	100%
F4	6.48E-07	8.88E-16	100%
F7	-1.26E+04	-1.26E+04	100%
F10	6.74E+00	9.75E-07	60%
F11	9.32E-07	7.87E-07	100%

Table 3 The results for the 6 tested functions, each with 50 variables.

	PSO		
function	f_{av}	f_b	Sucess Rate
F1	9.32E-07	7.46E-07	100%
F3	2.18E+04	9.98E-07	10%
F4	1.89E+01	9.63E-07	3.33%
F7	-1.06E+04	-1.33E+04	0
F10	1.59E+08	1.90E+07	0
F11	1.81E+01	1.49E+01	0
	MPSO		
function	f_{av}	f_b	Sucess Rate
F1	7.22E-07	0.00E+00	100%
F3	9.99E-07	9.95E-07	100%
F4	8.88E-16	8.88E-16	100%
F7	-2.09E+04	-2.09E+04	100%
F10	1.85E+01	9.75E-07	36.66%
F11	9.54E-07	7.74E-07	100%

number of maximum generations, we then succeed in obtaining a solution that is close to the exact solution.

When the number of variables is equal to 100, the optimization results shown in Table 4 indicate that, for this case, the PSO presents a success rate of zero for 4 of the 6 tested functions and even for the remaining two gives a success rates of less than 20%. With respect to the MPSO for this case, the success rates for the functions F5 drop to between 10% and 20% as the number of variables is increased from 50 to 100; however, the MPSO still performs with a 100% success rate for the remaining functions.

According to the above optimization results for a different number of variables, the MPSO performs perfectly, with a success rate 100%, for the separable functions and is never affected by the number of

Table 4 The results for the 11 tested functions, each with 100 variables.

	PSO		
function	f_{av}	f_b	Sucess Rate
F1	1.71E+04	9.97E-07	16.66%
F3	5.15E+07	9.99E-07	6.66%
F4	1.99E+01	1.99E+01	0
F7	-2.13E+04	-2.48E+04	0
F10	6.57E+08	2.52E+08	0
F11	2.00E+01	1.89E+01	0
	MPSO		
function	f_{av}	f_b	Sucess Rate
F1	0.00E+00	0.00E+00	100%
F3	1.00E-06	9.97E-07	100%
F4	8.88E-16	8.88E-16	100%
F7	-4.18E+04	-4.18E+04	100%
F10	3.70E+01	9.97E-07	13.33%
F11	9.71E-07	8.63E-07	100%

variables. Furthermore, for the multiple modal benchmark functions, such as F3, F4 and F6, the particles that are found to be stagnant in a certain place during the application of the PSO are driven by the MPSO′s strategy of introducing the three position vectors to replace the three particles with the worst fitness.

5. Conclusions

In conclusion, we developed a modified PSO (MPSO) algorithm by leveraging successive interval search method and replacing the three worst fitness particles with three creative position vectors. This MPSO algorithm was shown to be able to overcome the premature convergence situation that occurs when a problem with a large number of variables and (or) multiple optima is solved.

Our results showed that our MPSO algorithm successfully secure a solution that is close to the exact solution for 5 of the 6 benchmark functions, except for the randomly shifted Rosenbrock functions. By increasing the maximum number of generations for our MPSO algorithms, we showed that all benchmark functions can be solved. In comparison to the PSO algorithm utilizing three different numbers of variables, it failed to obtain a solution that is close enough to the exact solution for all of the tested functions except for the Sphere function with 30 variables.

Acknowledgement

We are grateful to National Science of Council of Taiwan for the financial support to the study with a grant number of NSC 99-2221-E-019-052.

References

[1] Kennedy, J. and R.C. Eberhart, Particle Swarm Optimization, Proc. IEEE International Conf. Neural Networks, **4**, 1942-1948 (1995).

[2] Suganthan, P.N., Particle Swarm Optimizer with Neighborhood Operator, Proc. Congress on Evolutionary Computation, **3**, 1958-1962 (1999).

[3] Ratnaweera, A., S.K. Halgamuge and H.C. Watson, Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients, Evolutionary Computation, IEEE Transactions, **8**, 240-255 (2004).

[4] Chatterjee, A. and P. Siarry, Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization, Computers and Operations Research, **33**, 859-871 (2004).

[5] Liu, B., L. Wang, Y.H. Jin, F. Tang and D.X. Huang, Improved Particle Swarm Optimization Combined with Chaos, Chaos, Solitons and Fractals, **25**, 1261-1271 (2005).

[6] Bin, J., L. Zhigang and G. Xingsheng, A Dynamic Inertia Weight Particle Swarm Optimization Algorithm, Chaos, Solutions and Fractals **37**, 698-705 (2006).

[7] Yang X., J. Yuan J. Yuan and H. Mao, A Modified Particle Swarm Optimizer with Dynamic Adaptation, Applied Mathematics and Computation, **189**, 1205-1213 (2007).

[8] Xiang, T., K.W. Wong and X. Liao, A Novel Particle Swarm Optimizer with Time-delay, Applied Mathematics and Computation, **186**, 789-793 (2006).

[9] Fan, S.K.S. and E. Zahara, A Hybrid Simplex Search and Particle Swarm Optimization for Unconstrained Optimization, European Journal of Operational Research, **181**, 527-548 (2006).

[10] Yeh, Wei-Chang, A simple Hybrid Particle Swarm Optimization, Advances in Evolutionary Algorithm, Book Edited by Witold Kosinski, ISBN 978-953-7619-11-4, (2008).

[11] Jasbir S. Arora, Introduction to Optimum Design, McGraw Hill, Inc. ISBN. 0-07-100123-9, (1989).

[12] Zhenyu Yang, Ke Tang, Xin Yao, Large scale evolutionary optimization using cooperative coevolution, Information Sciences **178**, 2985-2999 (2008).

Hsin Chuan Kuo received, in 1994, the PhD degree in optimization design on ship structure from the department of Engineering Science and Ocean Engineering, National Taiwan University. He is currently an associate professor in National Taiwan Ocean University. His research interests are in the areas of network architecture, Taguchi method, and optimization techniques.

Jeun-Len Wu received the Ph.D. degree from the Department of Engineering Science, Virginia Polytechnic Institute & State University. He is currently a professor in the Department of Systems Engineering & Naval Architecture in National Taiwan Ocean University. Furthermore, he is also the Vice President of the University in charge of college administration affairs. His research interests are in the areas of Flow Stability, Computation Fluid Dynamics, Computation Evolution Method and Optimization Designs.

Ching-Hai Lin received his MS degree in Electrical Engineering from National Taipei University of Technology, Taipei, Taiwan. He is currently working toward his Ph.D. degree in Systems Engineering and Naval Architecture Department at National Taiwan Ocean University, Keelung, Taiwan. His research interests include system engineering, software computing, and optimization algorithms.