2010

A particle swarm optimization algorithm based on orthogonal design

Jie Yang
University of Wollongong, jy962@uow.edu.au

Abdesselam Bouzerdoum
University of Wollongong, bouzer@uow.edu.au

Son Lam Phung
University of Wollongong, phung@uow.edu.au

Publication Details

Yang, J., Bouzerdoum, A. & Phung, S. (2010). A particle swarm optimization algorithm based on orthogonal design. WCCI 2010 IEEE World Congress on Computational Intelligence (pp. 593-599). USA: IEEE.

A particle swarm optimization algorithm based on orthogonal design

Abstract
The last decade has witnessed a great interest in using evolutionary algorithms, such as genetic algorithms, evolutionary strategies and particle swarm optimization (PSO), for multivariate optimization. This paper presents a hybrid algorithm for searching a complex domain space, by combining the PSO and orthogonal design. In the standard PSO, each particle focuses only on the error propagated back from the best particle, without "communicating" with other particles. In our approach, this limitation of the standard PSO is overcome by using a novel crossover operator based on orthogonal design. Furthermore, instead of the "generating-and-updating" model in the standard PSO, the elitism preservation strategy is applied to determine the possible movements of the candidate particles in the subsequent iterations. Experimental results demonstrate that our algorithm has a better performance compared to existing methods, including five PSO algorithms and three evolutionary algorithms.

Keywords
orthogonal, particle, design, swarm, algorithm, optimization

Disciplines
Physical Sciences and Mathematics

Publication Details
Yang, J., Bouzerdoum, A. & Phung, S. (2010). A particle swarm optimization algorithm based on orthogonal design. WCCI 2010 IEEE World Congress on Computational Intelligence (pp. 593-599). USA: IEEE.

A Particle Swarm Optimization Algorithm based on Orthogonal Design

Jie Yang, Abdesselam Bouzerdoum, *Senior Member, IEEE* and Son Lam Phung, *Member, IEEE*

***Abstract*— The last decade has witnessed a great interest in using evolutionary algorithms, such as genetic algorithms, evolutionary strategies and particle swarm optimization (PSO), for multivariate optimization. This paper presents a hybrid algorithm for searching a complex domain space, by combining the PSO and orthogonal design. In the standard PSO, each particle focuses only on the error propagated back from the best particle, without "communicating" with other particles. In our approach, this limitation of the standard PSO is overcome by using a novel crossover operator based on orthogonal design. Furthermore, instead of the "generating-and-updating" model in the standard PSO, the elitism preservation strategy is applied to determine the possible movements of the candidate particles in the subsequent iterations. Experimental results demonstrate that our algorithm has a better performance compared to existing methods, including five PSO algorithms and three evolutionary algorithms.**

I. Introduction

We consider the optimization problem in a high-dimensional space. Let $f(\boldsymbol{x})$ be the objective function of N variables, where $\boldsymbol{x} = (x_1, ..., x_N)$. The problem can be described mathematically as

$$\text{minimizing } f(\boldsymbol{x}) \text{ subject to } x_i \in \mathbf{E} \text{ for all } i \in N, \quad (1)$$

where $\mathbf{E}$ denotes the set of constraints on the variables x_i. A constraint could be the upper and lower limit of the variable, such as $a_i \leq x_i \leq b_i$.

There are several approaches to solving this optimization problem. The gradient-based approach starts with an initial estimate $\boldsymbol{x}_0$. At each iteration, it calculates the gradient of the objective function in order to find the next estimate:

$$\boldsymbol{x}_{\text{next}} = \boldsymbol{x} - \alpha \nabla f(\boldsymbol{x}), \quad (2)$$

where α is the learning rate. The gradient-based approach can be applied to only objective functions that are differentiable. It is sensitive to the initial point, and it requires large memory for gradient computation.

Another optimization approach uses evolutionary algorithms. It evaluates and updates a population of solutions iteratively. At each iteration, the optimal solutions are preserved according to the principle of "survival of the fittest". There are different strategies for the interaction within the population, including genetic algorithms, niche technology, ant colony algorithm and particle swarm optimization. This population-based approach is less sensitive to the initial conditions, and using a sufficient population size it can converge to a solution.

The authors are with the School of Electrical, Computer and Telecommunications Engineering, and ICT Research Institute, University of Wollongong, Australia. E-mail: {jy962, bouzer, phung}@uow.edu.au).

Among the evolutionary algorithms, particle swarm optimization is considered as a powerful and problem-independent method for multivariate optimization. It is inspired by the social behavior of animals such as bird flocking, and was introduced by Kennedy and Eberhart [1]. PSO begins with an initial population of randomly generated individuals or *particles*. It updates the particles and selects the one with the maximum fitness as the output. PSO differs from other evolutionary algorithms in the genetic operator that is used for generating the next population. In PSO, the "father" particles are updated according to the bias of their current positions and velocities.

However, PSO has been shown to have slow convergence and low solution accuracy, especially for problems involving a high-dimensional space. To overcome these limitations, several approaches have been developed [3]–[20], which can be divided into three main categories: the multi-swarm, multi-phase stategy, the information-based updating strategy, and the hybrid strategy. Yan et al. [6] partition the entire population into several sub-swarms and execute the PSO on each of them independently. Some rules for using the best particles to modify the position and velocity of particles are proposed in [9], [14]. Brits et al. [10] apply the concept of "niches" to PSO, in which different functions for "fitness sharing" and "crowding" are defined. Kai et al. [19] divide the optimization process into the "isolated", "communing" and "united" phases, and split the whole swarm into some sub-populations, called "tribes". Lovbjerg et al. [20] combine the traditional updating rules with the ideas of breeding and sub-populations.

The hybrid strategy combines the standard PSO with other optimization algorithms to achieve faster convergence and better solution accuracy. It uses a local search to complement the exploring ability of the standard PSO. Examples of hybrid strategy are listed in Table I.

TABLE I
Hybrid algorithms.

Algorithm	Reference
PSO + Genetic Algorithm	[3], [13], [18]
PSO + Nelder-Mead simplex search method	[4], [11]
PSO + Expectation Maximization	[5]
PSO + Ant Colony Algorithms	[7]
PSO + Chaos + Linear Interior Point method	[12]
PSO + Sequential Quadratic Programming technique	[15]
PSO + Chaos + Adaptive Inertia Weight Factor	[16]
PSO + Alternating Least Squares	[17]

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

We propose a particle swarm optimization approach based on orthogonal design (OD). In our approach, termed ODPSO, each particle in the swarm can be divided into several partial vectors where each of them acts as a factor in the orthogonal design. Orthogonal design is then employed to search the best scales among all the various combinations. Compared to previous OD-based methods [21]–[25], our proposed algorithm has the following properties.

1) While orthogonal design has been applied to determine the optimal parameters for the standard PSO [8], and to adjust particle velocity [23], we consider orthogonal design as a crossover operator which affect the particles directly. The proposed ODPSO evaluates all candidate particles generated by the OD operator and directly selects the particle with the best fitness to replace the worst particle in the previous swarm.

2) A fundamental advantage of our algorithm over other methods is that we adopt the crossover operator which involves more selected particles than just two parents. Theoretically, it is more effective to explore the search space with a multi-parent operator than the classical two-parent operator [21] [23]. As the diversity of recombination is enhanced, so are the solution accuracy and the convergence rate.

3) Most of classical PSO methods adopt the "generating-and-updating" strategy, in which each particle will move to the next position after the velocity is updated. This updating rule helps in maintaining the swarm diversity, but it slows down the convergence. The proposed approach improves convergence by using *elitism preservation strategy*, in which a particle is updated only if it fitness is guaranteed to improve.

The rest of the paper is organized as follows. Section II reviews the fundamentals of particle swarm optimization and orthogonal design. Section III describes the proposed ODPSO approach and discusses the OD-based operator and the elitism preservation strategy. Section IV presents experimental results and analysis of the proposed approach on several constrained multivariate optimization problems. Section V gives the conclusions.

II. Standard PSO and Orthogonal Design

This section provides some background for the standard PSO algorithm and the concept of orthogonal design.

A. Standard PSO

Particle swarm optimization is inspired by the behaviors of birds when they try to locate food [1]. In PSO, each candidate solution, also called a "particle", is composed of two parts: the position and the velocity. Like other evolutionary algorithms, PSO conducts a search based on the whole swarm. Let $\boldsymbol{x}_i$ be the position and $\boldsymbol{v}_i$ be the velocity of the i-th particle. Let $\boldsymbol{p}_i$ be the *best position* of the ith particle, and $\boldsymbol{g}$ be the *global best position* of the whole swarm. At each iteration, the position and the velocity of a particle are updated according to the values of $\boldsymbol{p}_i$ and $\boldsymbol{g}$:

$$\boldsymbol{v}_i^{t+1} = w\ \boldsymbol{v}_i^t + c_1\ r_1(\boldsymbol{p}_i - \boldsymbol{x}_i^t) + c_2\ r_2\ (\boldsymbol{g} - \boldsymbol{x}_i^t) \quad (3)$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \quad (4)$$

Here, w, c_1 and c_2 are acceleration factors and r_1 and r_2 are uniform random variables in the interval $[-1, 1]$.

B. Orthogonal design

Consider an experiment that involves some *factors*, each of which have several possible values called *levels*. Suppose that there are P factors, each factor has Q levels. The number of combinations is Q^P, and for large P and Q it is not practical to evaluate all combinations.

Orthogonal design has been developed as a mathematical tool to study multi-factor and multi-level problems. It aims to extract an orthogonal array L of M rows, where each row represents a combination to be evaluated. The array has three key properties.

1) During the experiment, the array represents a subset of M combinations, from all possible Q^P combinations. Computation is reduced considerably because $M << Q^P$.

2) Each column represents a factor. If some columns are deleted from the array, it means a smaller number of factors are considered.

3) The columns of the array are orthogonal to each other. The selected subset is scattered uniformly over the search space to ensure its diversity.

A simple but efficient method is proposed in [21] to generate an orthogonal array L where $M = Q \times Q$ and $P = Q + 1$. The steps of this method are shown in Algorithm 1.

input : The number of levels Q
output: An orthogonal array L

Calculate $M = Q \times Q$ and $P = Q + 1$.
Initialize an zero matrix L with M rows and P columns.
for $i = 1$ **to** M **do**
  $L_{i,1} = \mod(\lfloor (i-1)/q \rfloor, q)$
  $L_{i,2} = \mod(i-1, q)$
  for $j = 1$ **to** $P - 2$ **do**
    $L_{i,2+j} = \mod(L_{i,1} \times j + L_{i,2}, q)$
  end
end

Algorithm 1: Procedure for generating an orthogonal array L.

III. Proposed Method

This section describes the proposed particle swarm optimization based on orthogonal design. We present the OD-based operator and the updating strategy and then describe the main steps of the proposed ODPSO algorithm.

A. OD-based operator and updating strategy

Each solution to the optimization problem is represented as a particle. For an objective function with N variables, a particle is encoded in the form of

$$\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, \cdots, x_{i,N}], \quad i = 1, 2, \cdots, S \tag{5}$$

where S is the swarm size.

The standard PSO algorithm updates the current swarm by comparing with the global best particle $\boldsymbol{g}$ and the local best $\boldsymbol{p}_i$, as in (3) and (4). It lacks the interaction between neighboring particles and is therefore easily trapped into local minima [3]–[6]. One technique to address this problem is to employ the multi-parent crossover during the evolution. When applied to GAs, this technique has been shown to improve the convergence rate.

Given m particles, the question is how to execute the multi-parent crossover efficiently. Since each particle consists of N factors, there are m^N combinations. Consequently, the orthogonal design method is employed to select M (if a $L_M(Q^P)$ orthogonal array is considered, where $P = N$ and $Q = m$) representative sets of combinations to shorten the computational time. The procedure of OD-based crossover is detailed in Algorithm 2.

input : m particles $\boldsymbol{x}_{i,j}, i \in [1, m]$ and $j \in [1, N]$
output: A new set of m particles $\boldsymbol{p}_{i,j}$

Construct the orthogonal array $L_{m\times m}\left(m^{m+1}\right)$ using Algorithm 1.
Delete the last $(m + 1 - n)$ columns of $L_{m\times m}\left(m^{m+1}\right)$ to get $A = L_{m\times m}\left(m^n\right)$.
Generate m new particles:
for $i = 1$ **to** m **do**
  for $j = 1$ **to** N **do**
    $index = A_{i,j}$
    $\boldsymbol{p}_{i,j} = \boldsymbol{x}_{index,j}$
  end
end
Compute the fitness for all $\boldsymbol{p}_{i,j}$.
Mix $\boldsymbol{p}_{i,j}$ and $\boldsymbol{x}_{i,j}$ and rank all particles in the decreasing order of fitness
Select the top m particles as the output.

Algorithm 2: OD-based operator for m particles.

Remark 1: The OD-based operator behaves as the local search among the selected particles. As shown in Fig. 1, the crossover operator considers several particles simultaneously to search the best combination of particles. This OD-based operator uses several parent particles and scatters the candidate solutions uniformly over the feasible space.

B. Steps of OD-based PSO

To obtain a more precise solution compared to the standard PSO, the OD-based operator is employed. The elitism preservation strategy for upgrading the current swarm is proposed, in which the particle is updated only if its fitness is improved. The procedure for the OD-based PSO is shown in Algorithm 3. A convergence criterion or the maximum run can be used as the termination condition.

input : Swarm size S, Maximum iterations K
output: Best particle

Construct a random initial population $\boldsymbol{x}_i$, $i \in [1, S]$.
for $t = 1$ **to** K **do**
  for $i = 1$ **to** S **do**
    $\boldsymbol{v}_i^{t+1} = w\ \boldsymbol{v}_i^t + c_1\ r_1(\boldsymbol{p}_i - \boldsymbol{x}_i^t) + c_2\ r_2\ (\boldsymbol{g} - \boldsymbol{x}_i^t)$
    Apply the elitism preservation strategy:

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} & \text{if} \quad f(\boldsymbol{x}_i^t) \le f(\boldsymbol{x}_i^{t+1}), \\ \boldsymbol{x}_i^t & \text{otherwise.} \end{cases}$$

    Select m particles randomly and execute Algorithm 2.
    Set $\boldsymbol{p}_i = \boldsymbol{x}_i^{t+1}$ if $f(\boldsymbol{x}_i^{t+1}) < f(\boldsymbol{p}_i)$.
    Set $\boldsymbol{g} = \arg\min f(\boldsymbol{p}_i)$.
  end
  Check the termination condition.
end
Take the particle $\boldsymbol{g}$ as the output.

Algorithm 3: OD-based particle swarm optimization for multivariate optimization.

Remark 2: The convergence of ODPSO is guaranteed because of the elitism preservation strategy. A particle moves only if the movement will lower this objective function. The ODPSO works well if there exists a time function $\alpha(t)$ in $(0, 1]$ such that the objective function decays as $f(\boldsymbol{x}_i^{t+1}) \le \alpha(t) f(\boldsymbol{x}_i^t)$, for i in $[1, S]$.

IV. Experiment Results and Analysis

In this section, we analyze the proposed ODPSO algorithm in terms of solution accuracy and success rates, and the effect of the OD operator on optimization performance. The experiments are run on several well-known benchmark problems in optimization. These problems are listed Table II. They vary in terms of the number of local or global minima and the size of the search space. We also compare the proposed algorithm with five PSO methods and three evolutionary algorithms.

A. Comparison with other PSO methods

The ODPSO algorithm is tested on 23 benchmark problems. In these problems, the number of variables goes from

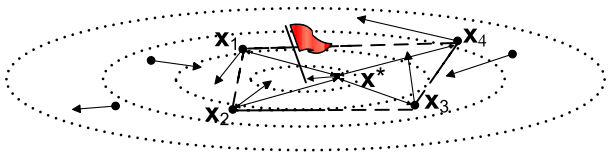


Fig. 1. Example of how the new particle $\boldsymbol{x}^*$ is generated from $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ using the OD crossover operator. The flag indicates the optimum particle, and the arrow represents particle velocity.

TABLE II

OBJECTIVE FUNCTIONS FOR ANALYSIS OF PSO ALGORITHMS.

Function	**Dimension** N	**Search space**
B2=$x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	2	$-100 \le x_i \le 100$
BR=$\left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	2	$-5 \le x_1 \le 10, 0 \le x_2 \le 15$
ES=$-\cos(x_1)\cos(x_2)\exp\left[-(x_1-\pi)^2 - (x_2-\pi)^2\right]$	2	$-100 \le x_i \le 100$
DJ=$x_1^2 + x_2^2 + x_3^2$	3	$-5.12 \le x_i \le 5.12$
VD=$\sum_{i=1}^{N}(x_i-1)^2 + \left[\sum_{i=1}^{N} i\,(x_i-1)\right]^2 + \left[\sum_{i=1}^{N} i\,(x_i-1)\right]^4$	4	$-100 \le x_i \le 100$
SP=$\sum_{i=1}^{N} x_i^2$	$10, 30$	$-100 \le x_i \le 100$
GR=$\sum_{i=1}^{N}\left(\frac{x_i^2}{4000}\right) - \prod_{j=1}^{N}\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$8, 10, 30, 50$	$-300 \le x_i \le 600$
RA=$\sum_{i=1}^{N}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$10, 20$	$-5.12 \le x_i \le 5.12$
RS=$\sum_{i=1}^{N}\left[100\left(x_i^2 - x_{i+1}^2\right) + (x_i-1)^2\right]$	$2, 4, 5, 10, 20$	$-5 \le x_i \le 10$
ZA=$\sum_{i=1}^{N} x_i^2 + \left(\sum_{j=1}^{N} 0.5jx_j\right)^2 + \left(\sum_{k=1}^{N} 0.5kx_k\right)^4$	$2, 5, 10$	$-5 \le x_i \le 10$
TR=$\sum_{i=1}^{N}\left[N - \sum_{j=1}^{N}\cos(x_j) + i\,(1-\cos(x_i)) - \sin(x_i)\right]^2$	$4, 8$	$-10 \le x_i \le 10$

2 to 30. The parameters we use for the ODPSO algorithm are listed in Table III. The values for w, c_1, and c_2 are recommended in [2]. We set parameter $V_{\max}$ to prevent the explosion of the generated particles.

TABLE III

PARAMETERS USED FOR THE ODPSO ALGORITHM.

Swarm size	Max iterations	Error goal	No. particles (m)
150	3000	$1e-100$	10
w	c_1	c_2	$V_{\max}$
$[0.4, 0.9]$	2.0	2.0	10

The particles are initialized randomly in the search space. The search terminates when the maximum number of iterations is reached, or the error between the best particle and the global mimimum is below 10^{-100}. The results of ODPSO are averaged over 30 independent runs. Five PSO-based algorithms are also applied on the same problems: GA-PSO [3], NM-PSO [4], IPSO [6], PSACO [7], and OPSO [23].

Table IV reports the mean error obtained by the proposed ODPSO and other PSO algorithms. The ODPSO algorithm outperforms five other PSO algorithms in 19 out of 23 test functions. For example, the ODPSO algorithm can locate the unique optimum perfectly (mean error of 0) in seven test functions, namely B2(2), BR(2), ES(2), VD(4), GR(8), RA(10) and RS(2), whereas other PSO algorithms cannot.

For other test functions DJ(3), SP(10), ZA(2) and ZA(5), the ODPSO algorithm achieves an average error of 5.9710^{-101}. The results seem to be reasonable because the parameter *Error Goal* is set to 10^{-100}. Furthermore, a better solution can be expected if this parameter is set lower. Although the results of ODPSO on three test functions GR(50), RA(20) and RS(20) are not the best, its performance can be improved by fine-tuning the parameter m; a detail discussion will be given in Section IV-C.

For six test functions GR(10), GR(30), GR(50), RA(20), RS(10), and RS(20), the success rate of the ODPSO algorithm is 90%, 80%, 64%, 68%, 88%, and 66%, respectively. For the other 17 test functions, ODPSO achieves a 100% success rate. The success rate is the percentage of runs that an algorithm reaches the global minimum.

In summary, the proposed ODPSO is very competitive compared with existing PSO algorithms, and the OD-based operator improves the efficiency of the original PSO.

B. Comparison with other EAs

In this section, the proposed method is compared with other evolutionary algorithms, on two optimization problems listed in Table V. The evolutionary algorithms are Niche Genetic Algorithm (NGA) [28], Ant Colony Algorithm (ACA) [29], and Genetic Algorithm with the Elitist Selection (GA-ES)[30]. The niche technique is considered as an effective strategy to maintain the population diversity. The ACA simulates the foraging behavior of ants. When the paths are encoded as the candidate solutions, the shortest path between the colony and a food source is regarded as the optimal solution. The main difference between GA-ES and the standard GA is the strategy for selecting the next population. The GA-ES uses the elitism preservation strategy instead of the classical roulette-wheel selection.

The parameters for ODPSO are the same as in Section IV-A and the parameters for other methods are listed in Table VI. In ACA, parameter P denotes the remaining pheromone used in each candidate solution, which depends on the corresponding fitness; parameter T is the attenuation coefficient for pheromone. The stopping criterion for NGA, ACA and GA-ES is reached after the maximum number of iterations.

Table VII presents the results of the ODPSO and three evolutionary algorithms, on the "Peaks" and "Himmelblau"

TABLE IV

COMPARISON OF ODPSO AND PSO-BASED ALGORITHMS IN TERMS OF THE MEAN ERROR.

Test Function	**Average Error**					
	ODPSO	GA-PSO [3]	NM-PSO [4]	IPSO [6]	PSACO [7]	OPSO [23]
B2(2)	**0**	1e-5	3.24e-10	-	5.55e-17	-
BR(2)	**0**	9e-5	3e-5	-	2.62e-13	-
ES(2)	**0**	3e-5	4e-5	-	0	-
DJ(3)	**5.68e-101**	4e-5	1.63e-13	-	7.69e-29	-
VD(4)	**0**	-	1.34e-9	-	-	-
SP(10)	6.39e-101	-	-	-	-	**0**
SP(30)	**3.20e-12**	-	2.76e-11	-	-	-
GR(8)	**0**	-	4.1e-4	-	6.23e-22	-
GR(10)	**7.40e-4**	-	-	-	-	1.00
GR(30)	**3.19e-10**	-	-	0.001	-	-
GR(50)	2.01e-8	-	**9.96e-12**	-	-	-
RA(10)	**0**	-	1.91e-11	0.800	-	2.25
RA(20)	10.35	-	-	**4.28**	-	-
RS(2)	**0**	6.4e-4	3e-5	-	1.72e-10	-
RS(4)	**2.88e-14**	-	1.71e-9	-	-	-
RS(5)	**4.14e-9**	1.3e-4	5.6e-3	-	1.85e-4	-
RS(10)	**0.052**	-	-	-	-	0.390
RS(20)	3.992	-	-	**0.477**	-	-
ZA(2)	**4.89e-101**	5e-5	3e-5	-	5.71e-27	-
ZA(5)	**6.89e-101**	0	2.6e-4	-	3.64e-17	-
ZA(10)	4.68e-85	**0**	-	-	2.0e-8	-
TR(4)	**6.63e-5**	-	7.57e-5	-	-	-
TR(8)	**2.73e-5**	-	3.03e-5	-	-	-

[1] The best result for each problem is highlighted in **bold font**.
[2] Some comparison results are not available in the published papers. They are indicated by the "-" symbol.

TABLE V

TWO OPTIMIZATION PROBLEMS: PEAKS AND HIMMELBLAU [27].

Test Function	**Search Space**
$f_1 = 3(1-x_1)^2 \times \exp\left[-x_1^2-(x_2+1)^2\right]$ $-10\left(x_1/5 - x_1^3 - x_2^5\right) \times \exp\left(-x_1^2-x_2^2\right)$ $-(1/3)\times \exp\left[-(x_1+1)^2 - x_2^2\right]$	$-30 \le x_1, x_2 \le 30$ One global minimum Two local minima
$f_2 = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$ $+0.1 \times \left[(x_1-3)^2 + (x_2-2)^2\right]$	$-6 \le x_1, x_2 \le 6$ One global minimum Three local minima

TABLE VI

PARAMETERS USED BY THE ***NGA***, ***ACA***, AND ***GA-ES*** APPROACH

Algorithm	Description
NGA	PopSize= 150, Iterations= 3000, $P_c = 0.6$ $P_m = 0.05$, NicheSize= 15
ACA	PopSize= 150, Iterations= 3000 $P = \exp(-\text{fitness})$, $T = 0.9$
GA-ES	PopSize= 150, Iterations= 3000 $P_c = 0.6$, $P_m = 0.05$

problems over 30 independent runs. In terms of convergence, for both problems the ODPSO converges to the global optima in all runs, i.e. a success rate of 100%. In comparison, the NGA, ACA and GA-ES have much lower success rates. For the "Peaks" problem, the success rate for NGA, ACA and GA-ES is 73.3%, 56.7% and 76.7%, respectively. For the "Himmelblau" problem, the success rate for NGA, ACA and GA-ES is 9.1%, 3.3% and 80.0%, respectively.

In terms of computation time, ODPSO is slower than ACA, and faster than NGA and GA-ES. NGA needs to partition the population into several sub-niches, which is time-consuming since it must process all individuals in each iteration. Although the elitism preservation strategy in GA-ES ensures that the chromosome is updated only if its fitness is improved, GA-ES lacks additional operators (such as the OD-based crossover in ODPSO) to explore the better solution.

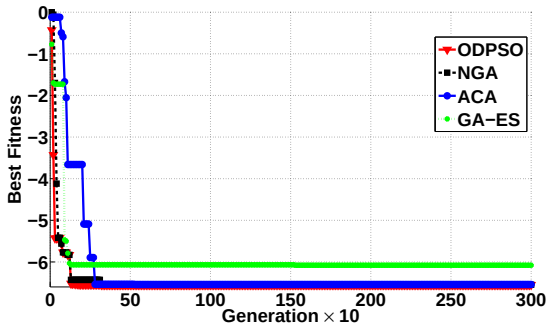
Figures 2 shows the best fitness at different iterations of the four methods (ODPSO, NGA, ACA and GA-ES), for one run in terms of the "Peaks" test function. As what we can see, the ODPSO method finds a lower minimum at each iteration, compared to the NGA, ACA, and GA-ES algorithms.

On the "Peaks" problem, after 370 iterations, the local minima achieved by the four algorithms are: ODPSO=-6.550, NGA=-6.5479, ACA=-6.5374, and GA-ES=-6.07. However, except for the proposed ODPSO, all

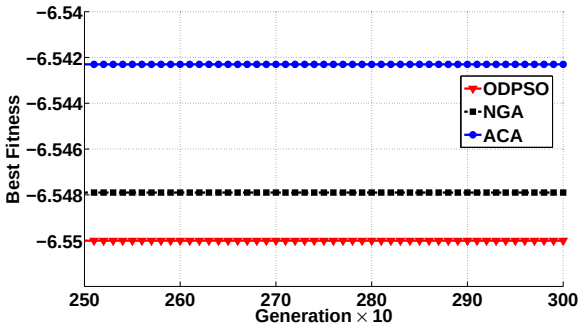
TABLE VII

COMPARISON OF ODPSO AND EVOLUTIONARY ALGORITHMS ON "PEAKS" AND "HIMMELBLAU" PROBLEMS.

	Peaks test function			
	ODPSO	NGA	ACA	GA-ES
Average minima	**-6.5511**	-6.5506	-6.5497	-6.5510
Computation time(s)	11.389	77.911	**2.293**	23.423
Success rate	**100%**	73.3%	56.7%	76.7%
	Himmelblau test function			
	ODPSO	NGA	ACA	GA-ES
Average minima	**0.0000**	0.0252	0.0771	1.590e-15
Computation time(s)	5.363	73.694	**0.398**	23.129
Success rate	**100%**	9.1%	3.3%	80.0%



(a) Computational results of "Peaks" function over 3000 iterations;



(b) Details about ODPSO, NGA and ACA in last 300 iterations from (a)

Fig. 2. Evolutionary curves of ODPSO, NGA, ACA, and GA-ES on the "Peaks" test function.

other algorithms are trapped in the local minima and the fitness does not improve after 370 iterations.

We skip the evolutionary curves obtained from the "Himmelblau" problem, but a similar phenomenon is observed, where NGA, ACA and GA-ES are trapped into the local optimal of 0.05, 0.0035 and $6.616e-015$, respectively. In comparison, the ODPSO only requires 1030 iterations to obtain the global minimum.

C. Sensitivity analysis of OD-based operator

In this section, we analyze the effect of OD-based operator on performance of ODSPO algorithm. Three test functions are used: GR(50), RA(20) and RS(20). Note that the ODPSO was observed not to perform well on these functions (see Section IV-A). We experiment with different values of m for the OD-based operator: $m = S/15$, $m = 2S/15$ and $m = 3S/15$.

Table VIII shows the performance of the ODPSO for 30 independent runs. It can be observed that with a higher m, the performance of ODPSO is improved. For all three test functions, the average error for $m = 30$ is much smaller than that for $m = 10$. The average error of ODPSO when $m = 30$ is lower than the average error of other PSO algorithms, listed in Table IV. When $m = 10$, the success rate of ODPSO for the three test functions is 64%, 68% and 66%, respectively. In comparison, when $m = 30$, the success rate of ODPSO increases to 89%, 90% and 88%, respectively. When more particles are used (by increasing m), the diversity of the candidate solutions generated by the OD operator is enhanced and the ODPSO is more likely to locate a better solution. Furthermore, the elitism preservation strategy used in the ODPSO improves convergence because new particles with lower fitness are discarded.

Table VIII shows that increasing m will lead to longer processing time for the OD-based operator. The value $m = 2S/15$ seems to have a good tradeoff between the computation time and solution accuracy.

V. CONCLUSIONS

Designing a practical and robust operator to enhance the performance of PSO is a significant and promising research direction in Evolutionary Computation. In this paper, a novel PSO algorithm (ODPSO) for multivariate optimizations is proposed. The main difference between our approach and the existing work is that we introduce a crossover operator based on the orthogonal design and an updating strategy based on elitism preservation.

The orthogonal design method has been proven that it can generate the candidate samples uniformly over the search space and select a representatives subset. Thus the proposed OD operator can be regarded as the a local search around the selected particles. It can help in exchanging the information among different particles as well as maintaining the diversity of the swarm. Instead of the the classical "generating-and-move" rule, we propose an updating strategy in which the particle will only move the new position if it has a better fitness. This updating strategy has been shown to result in better convergence.

Evaluated on several benchmark optimization problems, the proposed ODPSO is shown to outperform several other PSO algorithms and evolutionary algorithms in terms of solution accuracy and success rate. Our experiments indicate that using more particles for the OD operator will produce a better solution to the optimization problem, albeit there is a tradeoff between computation time and solution accuracy.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. IEEE International Conference on Neural Networks*, 4, Nov/Dec. 1995, pp. 1942–1948.

[2] Y. Shi and R.C. Eberhart, "A modified particle swarm optimizer," *Proc. Proceedings of the IEEE Congress on Evolutionary Computation*, Piscataway, NJ, 1998, pp. 69–73.

TABLE VIII

AVERAGE PERFORMANCES WITH DIFFERENT PARAMETERS FOR THE OD-BASED OPERATOR.

Function	$GR(50)$			$RA(20)$			$RS(20)$		
	$m = 10$	$m = 20$	$m = 30$	$m = 10$	$m = 20$	$m = 30$	$m = 10$	$m = 20$	$m = 30$
Average error	2.012e−8	8.33e−16	**1.99e-16**	10.348	3.980	**2.985**	3.992	0.264	**0.249**
Success rate	64	81	**89**	68	82	**90**	66	86	**88**
Computation time (s)	**35.84**	54.37	98.69	**6.60**	11.09	19.98	**3.86**	7.09	15.10

[3] K. Yi-Tung and Z. Erwie, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," *Applied Soft Computing*, vol. 8, pp. 849–857, 2008.

[4] S. F. Shu-Kai and Z. Erwie, "A hybrid simplex search and particle swarm optimization for unconstrained optimization," *European Journal of Operational Research*, vol. 181, pp. 527–548, 2007.

[5] S. F. Shu-Kai and L. Yen, "A multi-level thresholding approach using a hybrid optima estimation algorithm," *Pattern Recognition Letters*, vol. 28, pp. 662–669, 2007.

[6] Y. Jiang, H. Tiesong, and W. Xianing, "An improved particle swarm optimization algorithm," *Applied Mathematics and Computation*, vol. 193, pp. 231–239, 2007.

[7] P.S. Shelokar, P. Siarry , and V.K. Jayaraman, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization," *Applied Mathematics and Computation*, vol. 188, pp. 129–142, 2007.

[8] K. Chia-Nan, C. Ying-Pin, and W. Chia-Ju, "An orthogonal-array-based particle swarm optimizer with nonlinear time-varying evolution," *Applied Mathematics and Computation*, vol. 191, pp. 272–279, 2007.

[9] H. Qi, L.M. Ruan, M. Shi, W. An, H.P. Tan, "Application of multi-phase particle swarm optimization technique to inverse radiation problem," *Journal of Quantitative Spectroscopy & Radiative Transfer*, vol. 109, pp. 476–493, 2008.

[10] R. Brits, A.P. Engelbrecht , F. van den Bergh, "Locating multiple optima using particle swarm optimization," *Applied Mathematics and Computation*, vol. 189, pp. 1859–1883, 2007.

[11] Z. Erwie, S. F. Shu-Kai, T. Du-Ming, "Optimal multi-thresholding using a hybrid optimization approach," *Pattern Recognition Letters*, vol. 26, pp. 1082–1095, 2005.

[12] J. Chuanwen, E. Bompard, "A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimization," *Mathematics and Computers in Simulation*, vol. 68, pp. 57–65, 2005.

[13] L. Zne-Jung, "A novel hybrid algorithm for function approximation," *Expert Systems with Applications*, vol. 34, pp. 384–390, 2008.

[14] B. Zhao, C.X. Guo , B.R. Bai, Y.J. Cao, "An improved particle swarm optimization algorithm for unit commitment," *Electrical Power and Energy Systems*, vol. 28, pp. 482–490, 2006.

[15] T. Aruldoss Albert Victoire, A. Ebenezer Jeyakumar, "Hybrid PSO–SQP for economic dispatch with valve-point effect," *Electric Power Systems Research*, vol. 71, pp. 51–59, 2004.

[16] L. Bo,W. Ling, J. Yi-Hui, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons and Fractals*, vol. 25, pp. 1261–1271, 2005.

[17] S. Hideyuki,J. Jian-Hui, I. Makio, "Self-modeling curve resolution (SMCR) by particle swarm optimization (PSO)," *Analytica Chimica Acta*, vol. 595, pp. 275–281, 2007.

[18] C. Xindi, Z. Nian, V. Ganesh, "Time series prediction with recurrent neural networks trained by a hybrid PSO-EA algorithm," *Neurocomputing*, vol. 70, pp. 2342–2353, 2007.

[19] C. Kai, L. Tonghua, C. Tongcheng, "Tribe-PSO: A novel global optimization algorithm and its application in molecular docking," *Chemo metrics and Intelligent Laboratory Systems*, vol. 82, pp. 248–259, 2006.

[20] M. Lovbjerg, T.K Rasmussen, T. Krink, "Hybrid particle swarm optimiser with breeding and subpopulations," *Proc. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, San Francisco, USA, July. 2001

[21] Y. Wing-Leung, W. Yuping, "An orthogonal genetic algorithm with quantization for global numerical Optimizationg," *IEEE Transactions on Evolutionary Computation*, vol. 5(1), pp. 40–53, 2001.

[22] G. Wenyin,C. Zhihua, and L. Charles, "ODE: A fast and robust differential evolution based on orthogonal design," *Proc. 19th Australian Joint Conference on Artificial Intelligence*, Hobart, Australia, December, pp: 709–718, 2006

[23] H. Shinn-Ying, L. Hung-Sui, and L. Weei-Hurng, etal, "OPSO: Orthogonal particle swarm optimization and its application to task assignment Problems," *IEEE transactions on Systems, Man, And Cybernetics, Part A: SYSTEMS AND HUMANS*, vol. 38, pp. 288–298, 2008.

[24] Z. Sanyou, S. Hui, K. Lishan, D. Lixin, "Orthogonal dynamic hill climbing algorithm: ODHC," *Evolutionary Computation in Dynamic and Uncertain Environments*, pp. 79-104, 2007.

[25] L. Kwon-Hee, Y. Jeong-Wook, P. Joon-Seong, etal, "An optimization algorithm using orthogonal arrays in discrete design space for structures," *Finite Elements in Analysis and Design*, vol. 40, pp. 121–135, 2003.

[26] C. Ching-Shui, "Orthogonal arrays with variable numbers of symbols," *The Annals of Statistics*, vol. 8(2), pp. 447–453, 1980.

[27] K. Deb, *Optimization for engineering design: Algorithms and examples*, New Delhi: Prentice-Hall of India, 1995.

[28] T. G. Vladimir, S. Hiroshi, T. I. Georgy, "A method to improve multiobjective genetic algorithm optimization of a self-fuel-providing LMFBR by niche induction among nondominated solutions," *Annals of Nuclear Energy*, vol. 27, pp. 397–410, 2000.

[29] D. Marco, B. Christian, "Ant colony optimization theory: A survey, Theoretical Computer Science," *Annals of Nuclear Energy*, vol. 344, pp. 243–278, 2005.

[30] R. Zi-Wu, S. Ye, C. Jun-Feng, "Hybrid simplex-improved genetic algorithm for global numerical optimization," *Acta Automatica Sinica*, vol. 33, pp. 91–95, 2007.