OPSO: Orthogonal Particle Swarm Optimization and Its Application to Task Assignment Problems

Shinn-Ying Ho, *Member, IEEE*, Hung-Sui Lin, Weei-Hurng Liauh, and Shinn-Jang Ho

***Abstract*—This paper proposes a novel variant of particle swarm optimization (PSO), named orthogonal PSO (OPSO), for solving intractable large parameter optimization problems. The standard version of PSO is associated with the lack of a mechanism responsible for the process of high-dimensional vector spaces. The high performance of OPSO arises mainly from a novel move behavior using an intelligent move mechanism (IMM) which applies orthogonal experimental design to adjust a velocity for each particle by using a systematic reasoning method instead of the conventional generate-and-go method. The IMM uses a divide-and-conquer approach to cope with the curse of dimensionality in determining the next move of particles. It is shown empirically that the OPSO performs well in solving parametric benchmark functions and a task assignment problem which is NP-complete compared with the standard PSO with the conventional move behavior. The OPSO with IMM is more specialized than the PSO and performs well on large-scale parameter optimization problems with few interactions between variables.**

***Index Terms*—Orthogonal experimental design (OED), orthogonal PSO (OPSO), particle swarm optimization (PSO), task assignment.**

I. Introduction

PARTICLE swarm optimization (PSO) is one of the evolutionary computation techniques. It is a population-based search algorithm that exploits a population of individuals to probe promising regions of the search space. In this context, the population is called a swarm, and the individuals are called particles. Each particle moves with an adaptable velocity within the search space and retains in its memory the best position it ever encountered. The standard version of PSO is briefly described here [1]–[3].

Assume a D-dimensional search space $S \subset R^D$ and a swarm consisting of Np particles. The position $\mathbf{X} = [x_1, x_2, \ldots, x_D]^{\mathrm{T}}$ of a particle in the search space S is treated as a candidate solution. The current position of the ith particle is a D-dimensional vector $\mathbf{X}_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]^{\mathrm{T}} \in \mathrm{S}$. The velocity of this particle is also a D-dimensional vector $\mathbf{V}_i = [\nu_{i1}, \nu_{i2}, \ldots, \nu_{iD}]^{\mathrm{T}} \in \mathrm{S}$. The best position encountered by the ith particle is denoted as $\mathbf{P}_i = [p_{i1}, p_{i2}, \ldots, p_{iD}]^{\mathrm{T}} \in \mathrm{S}$. Assume g to be the index of the particle that attained the best position found by all particles in the ith particle's neighborhood. The swarm is manipulated in some form resembling the following equations:

$$\nu_{id}(t+1) = w\nu_{id}(t) + c_1 \cdot \mathrm{rand}() \cdot (p_{id}(t) - x_{id}(t)) + c_2 \cdot \mathrm{rand}() \cdot (p_{gd}(t) - x_{id}(t)) \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + \nu_{id}(t+1) \quad (2)$$

where $i = 1, 2, \ldots, \mathrm{Np}$ is the particle's index, $d = 1, 2, \ldots, D$ is the dimension index, and $t = 1, 2, \ldots$ indicates the iteration number. The variable w is a parameter called inertia weight, c_1 and c_2 are positive constants, which are referred to as cognitive and social parameters, respectively, and $\mathrm{rand}()$ is a function which generates a random number that is uniformly distributed within the interval [0,1]. There are two commonly used versions of PSO, i.e., the global version and the local version. In the local version of PSO, each particle's neighborhood includes a limited number of social neighbors, whereas in the global version of PSO, it includes all the particles in the swarm.

The inertia weight w plays the role of balancing the global and local searches. It can be a positive constant (e.g., 0.9) or even a positive linear or nonlinear function of time [2], [3]. Although sometimes proper fine-tuning of the parameters c_1 and c_2 leads to an improved performance, an extended study of the cognitive and social parameters is proposed in [4], and $c_1 = c_2 = 2$ were proposed as default values.

Recently, Clerc and Kennedy [7] indicated that a constriction factor χ may help ensure convergence. The new velocity of a particle is manipulated by the following equation:

$$\nu_{id}(t+1) = \chi\left[\nu_{id}(t) + d_1 \cdot \mathrm{rand}() \cdot (p_{id}(t) - x_{id}(t)) + d_2 \cdot \mathrm{rand}() \cdot (p_{gd}(t) - x_{id}(t))\right]. \quad (3)$$

The typical value of χ is

$$\chi = \frac{2\kappa}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}, \quad \text{where} \quad \phi = d_1 + d_2 > 4, \quad \text{and} \quad \kappa = 1. \quad (4)$$

Generalized models and techniques for controlling the convergence properties of PSO by fine-tuning its parameters are

Manuscript received September 26, 2004; revised June 5, 2005 and December 29, 2005. This paper was recommended by Associate Editor G. C. Calafiore.

S.-Y. Ho is with the Department of Biological Science and Technology and the Institute of Bioinformatics, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C.

H.-S. Lin and W.-H. Liauh are with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan, R.O.C.

S.-J. Ho is with the Department of Automation Engineering, National Formosa University, Yunlin 632, Taiwan, R.O.C. (e-mail: sjho@nfu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2007.914796

analyzed in [7]. Generally, $d_1 = d_2 = 2.05$ are often used, and thus, $\chi = 0.72984$ from (4). Therefore, (3) is equivalent to (1) with $w = 0.72984$ and $c_1 = c_2 = 1.4962$.

To prevent an explosion of the velocity, the usual method is simply to define a parameter $V_{\max}$ and prevent the velocity from exceeding it on each dimension d for particle i

$$\begin{aligned} &\text{If} \quad \nu_{id} > V_{\max}, \quad \text{then} \quad \nu_{id} = V_{\max}; \\ &\qquad \text{else if} \quad \nu_{id} < -V_{\max}, \quad \text{then} \quad \nu_{id} = -V_{\max}. \end{aligned} \tag{5}$$

The initialization of the swarm and velocities is usually performed randomly in the search space, although more sophisticated initialization techniques can enhance the overall performance of PSO [5]. The initial velocities are often distributed randomly over $[-V_{\max}, V_{\max}]$. According to the suggestion by Eberhart and Shi [6], it is good to limit the maximum velocity $V_{\max}$ to the upper value of the dynamic range of search.

Many intractable engineering problems, such as task assignment problem [8]–[10], are characterized by a nonlinear multimodal search space and a large number D of system parameters. For a high-dimensional vector used as a potential solution, there is a large probability that some partial vectors move closer to the optimal solution, whereas others move away from the optimal solution. Because the velocity is adjusted on a full D-dimensional vector, a drawback of the standard PSO is associated with the lack of a mechanism responsible for the process of the high-dimensional vector space. van den Bergh and Engelbrecht [11] proposed a cooperative particle swarm optimizer which uses multiple swarms to optimize different components of the solution vector cooperatively.

In this paper, we propose an entirely different approach to improve the PSO performance for overcoming curse of dimensionality. Although several improved versions of PSO have been proposed, many of which have been based on the original move behavior in (1). Different from the improvement of convergence and regulation of tradeoff between the local and global exploration abilities of swarm by tuning control parameters, the proposed orthogonal PSO (OPSO) employs a novel move behavior to significantly improve the performance of PSO. The OPSO uses an intelligent move mechanism (IMM) which applies orthogonal experimental design (OED) [12]–[14] to adjust a velocity for each particle. Each particle using IMM can adjust its flying velocity from the viewpoint of individual parameters (partial vectors) rather than the full D-dimensional vector.

The OED-based IMM uses a divide-and-conquer technique to efficiently determine the next move of a particle. Generally, a D-dimensional vector of the next move is divided into $N(\leq D)$ partial vectors. The IMM spends at most $2N$ objective function evaluations to reason a potentially good solution to be the next move consisting of N good partial vectors. The IMM generates the next move by using a systematic reasoning method instead of the generate-and-go method of the conventional PSO, resulting in economically obtaining an accurate solution to the intractable engineering problems.

OED with both orthogonal array (OA) and factor analysis is a representative method of quality control, which is also an efficient technique in the Taguchi method [15]. Tsai *et al.* [16] proposed a hybrid Taguchi–genetic algorithm (GA) for global numerical optimization where the Taguchi method is inserted between the crossover and mutation operations of a GA. Tanaka [17] proposed an orthogonal design algorithm using GA-encoding and OED with no recombination or mutation. Zhang and Leung [18] proposed an orthogonal GA (OGA). Leung and Wang [19] proposed an improved OGA with quantization (OGA/Q) using an OA-based initial population for global numerical optimization. Both the OGA and the OGA/Q use OA without factor analysis. Ho *et al.* [20]–[22] proposed population-based evolutionary algorithms with an OED-based recombination to efficiently solve large parameter optimization problems. In addition, the OED also performs well in cooperating with the simulated annealing (SA) [23], [24]. The original contribution of this paper is to apply the OED to the PSO rather than the GA and the SA. The resultant OPSO is a population-based algorithm with no recombination that is completely new with respect to published research works.

The task assignment problem, which is known to be NP-complete [25], is that of assigning tasks of a program among different processors of a distributed computer system in order to reduce the program turnaround time and to increase the system throughput [8]–[10]. In this paper, the local version of PSO with a constriction factor for parametric optimization functions and the global version of PSO for the task assignment problem [8] are selected for comparisons with the OPSO. Performance comparisons show that the OPSO with IMM is better than the PSO with the conventional move behavior in a limited amount of computation time using these test problems.

The rest of this paper is organized as follows. Section II presents the OED used for the IMM of OPSO. Section III gives the proposed OPSO with IMM. The performance comparisons of OPSO using the parametric optimization functions are presented in Section IV. Section V describes the application of OPSO to the task assignment problems. Finally, Section VI concludes this paper.

II. Orthogonal Experimental Design

For self-completeness, the concise description of the used OED is given. More information for the efficient use of OED can be found in [20]–[24].

A. Concepts of OED

An efficient way to study the effect of several factors simultaneously is to use the OED with both the OA and the factor analysis. The factors are the variables (parameters), which affect response variables, and a setting (or a discriminative value) of a factor is regarded as a level of the factor. A "complete factorial" experiment would make measurements at each of all possible level combinations. However, the number of level combinations is often so large that this is impractical, and a subset of level combinations must be judiciously selected to be used, resulting in a "fractional factorial" experiment [12]–[14]. The OED utilizes properties of the fractional factorial experiments

to efficiently determine the best combination of factor levels to be used in design problems.

The OED can provide near-optimal quality characteristics for a specific objective. Furthermore, there is a large saving in the experimental effort. The OED uses well planned and controlled experiments in which certain factors are systematically set and modified, and then, the main effect of factors on the response can be observed. The OED specifies the procedure of drawing a representative sample of experiments with the intention of reaching a sound decision [15]. Therefore, the OED using the OA and the factor analysis is regarded as a systematic reasoning method.

B. Orthogonal Array

OA is a fractional factorial array, which assures a balanced comparison of levels of any factor. OA is an array of numbers arranged in rows and columns where each row represents the levels of factors in each combination, and each column represents a specific factor that can be changed from each combination. The term "main effect" designates the effect on response variables that one can trace to a design parameter. The array is called orthogonal because all columns can be evaluated independently of one another, and the main effect of one factor does not bother the estimation of the main effect of another factor [12], [13].

The used OA in IMM is described next. Let there be N factors with two levels each. The total number of level combinations is 2^N for a complete factorial experiment. To use an OA of N factors, we obtain an integer $M = 2^{\lceil \log_2(N+1) \rceil}$ where the bracket represents an upper ceiling operation, build an OA $L_M(2^{M-1})$ with M rows and $M-1$ columns, use the first N columns, and ignore the other $M-N-1$ columns. For example, if $N \in \{4, 5, 6, 7\}$, then $M = 8$, and $L_8(2^7)$ is used. The numbers one and two in each column indicate the levels of the factors. Each column has an equal number of ones and twos. The array is orthogonal when the four pairs, i.e., (1, 1), (1, 2), (2, 1), and (2, 2), appear the same number of times in any two columns.

The OA can reduce the number of level combinations for factor analysis. The number of OA combinations required to analyze all individual factors is only $M = O(N)$, where $N + 1 \leq M \leq 2N$. A simple algorithm for constructing the used two-level OAs can be found in [22]. After a proper tabulation of experimental results, the summarized data are analyzed using the factor analysis to determine the relative level effects of factors.

C. Factor Analysis

Factor analysis using the OA's tabulation of experimental results can allow the main effects to be rapidly estimated, without the fear of distortion of results by the effects of other factors. Factor analysis can evaluate the effects of individual factors on the evaluation function, rank the most effective factors, and determine the best level for each factor such that the evaluation function is optimized.

Let f_z denote a function value of the combination z, where $z = 1, \ldots, M$. Define the main effect of factor j with level k as S_{jk}, where $j = 1, \ldots, N$, and $k = 1, 2$

$$S_{jk} = \sum_{z=1}^{M} f_z \cdot F_z \tag{6}$$

where $F_z = 1$ if the level of factor j of combination z is k; otherwise, $F_z = 0$. Considering the case that the optimization function is to be maximized, the level 1 of factor j makes a better contribution to the function than the level 2 of factor j when $S_{j1} > S_{j2}$. If $S_{j1} < S_{j2}$, level 2 is better. On the contrary, if the function is to be minimized, level 2 (level 1) is better when $S_{j1} > S_{j2} (S_{j1} < S_{j2})$. If $S_{j1} = S_{j2}$, levels 1 and 2 have the same contribution. The main effect reveals the individual effect of a factor. The most effective factor j has the largest main effect difference $\mathrm{MED} = |S_{j1} - S_{j2}|$.

Note that the main effect holds only when no or weak interaction exists, and that makes the experiment meaningful. An actual experiment result is estimated based only on the factors with the major effect. The difference between the estimated and experimental results is the degree of interactions among factors. In order to achieve an effective design, experiments should be prepared so as to reduce the interactions among factors. For generalized optimization problems with parameters having more or less strong interactions, the OPSO can still work well because the evolution ability of the PSO can compensate the OED excluding the study of interactions.

After the better one of two levels of each factor is determined, an intelligent combination consisting of all factors with the better levels can be easily derived. The OED is effective for the development design of efficient search for the intelligent combination of factor levels, which can yield a high-quality objective function value, compared with all values of generated combinations.

III. Proposed Optimal PSO

IMM is the main phase in adjusting the particles' velocities of OPSO. The major concerns of efficiently using the IMM are the following: 1) how to encode system parameters into the position $\mathbf{X}$ of a particle and 2) how to effectively divide $\mathbf{X}$ into N partial vectors where each partial vector is treated as a factor of the OED. The representation $\mathbf{X}$ plays an important role in making the IMM efficient, as described in Section III-A. How to efficiently use the IMM involving the division of $\mathbf{X}$ is given in Section III-B. The proposed OPSO is given in Section III-C.

A. Particle Representation

A suitable way of encoding the system parameters into the position $\mathbf{X}$ of a particle is important in efficiently using the IMM. An experienced engineer of using IMM tries to use appropriate parameter transformation for handling constraints in order to confine searches within feasible regions. Considering the properties of OED, the main effects of individual factors are more accurate for unconstrained problems than constrained problems if the constraints are not properly handled. Therefore,

problem-specific particle representation and specialized operations are generally more efficient in solving the constrained problems than any other methods using penalty approaches. The necessity of maintaining feasibility can make the OED-based operation more efficient than the penalty approach [22].

To accurately estimate the main effect of OED, the position of a particle should be encoded so as to reduce the degree of epistasis among parameters x_i and maintain the feasibility of all conducted combinations. A proper encoding scheme of particles where the parameters with strong interactions (if prior knowledge is available) are encoded together would enhance the effectiveness of particle division. An illustrative example using an effective parameter transformation to reduce the degree of epistasis and confine searches within feasible regions for designing genetic-fuzzy systems can be found in [20]. Consequently, the experiments can be prepared in order to reduce the interactions among factors by properly assigning partial vectors to individual factors.

B. Intelligent Move Mechanism

For the position of each particle $\mathbf{X}_i(t) = [x_{i1}(t), \ldots, x_{iD}(t)]^{\mathrm{T}}$, the IMM generates two temporary moves $\mathbf{H} = [h_1, h_2, \ldots, h_D]^{\mathrm{T}}$ and $\mathbf{R} = [r_1, r_2, \ldots, r_D]^{\mathrm{T}}$ corresponding to the cognitive and social parts, respectively

$$h_d = x_{id}(t) + w\nu_{id}(t) + c_1 \cdot \mathrm{rand}() \cdot (p_{id}(t) - x_{id}(t)) \quad (7)$$

$$r_d = x_{id}(t) + w\nu_{id}(t) + c_2 \cdot \mathrm{rand}() \cdot (p_{gd}(t) - x_{id}(t)). \quad (8)$$

The IMM aims at efficiently combining good partial vectors of $\mathbf{H}$ and $\mathbf{R}$ to generate the next move $\mathbf{X}_i(t+1)$, as described here.

Divide the D-dimensional vector of $\mathbf{X}_i$ into N nonoverlapping partial vectors using the same division scheme for $\mathbf{H}$ and $\mathbf{R}$. The proper value of N is problem-dependent. The larger the value of N, the more efficient it is the IMM if the interactions among partial vectors are weak. If the existing interactions among partial vectors are not weak, the smaller the value of N, the more accurate it is the estimated main effect of individual partial vectors. Considering the tradeoff, an efficient division criterion is to minimize the interactions among partial vectors while maximizing the value of N. To efficiently use all columns of OA excluding the study of intractable interactions, the used OA is $L_{N+1}(2^N)$, and the largest value of N is equal to $2^{\lfloor \log_2(D+1) \rfloor} - 1$ where the bracket represents a lower ceiling operation. The $N-1$ cut points are randomly specified from the $D-1$ candidate cut points which separate individual parameters. Note that the parameter N at each call of the following IMM operation can be a constant or variable value. For example, a coarse-to-fine strategy using a variable value of N is sometimes more efficient [21] (for a more detailed study of efficiently assigning parameters to factors, please refer to the OED-based optimization algorithms with real-world applications [20]–[24]).

How to perform an IMM operation using $\mathbf{X}_i(t)$ with an objective function f is described as follows.

Step 1) Generate two temporary moves $\mathbf{H}$ and $\mathbf{R}$ for $\mathbf{X}_i(t)$ using (7) and (8).

Step 2) Divide each of $\mathbf{H}$ and $\mathbf{R}$ into N partial vectors where each partial vector is treated as a factor.

Step 3) Use the first N columns of an OA $L_M(2^{M-1})$, where $M = 2^{\lceil \log_2(N+1) \rceil}$.

Step 4) Let levels 1 and 2 of factor i represent the ith partial vectors of $\mathbf{H}$ and $\mathbf{R}$, respectively.

Step 5) Compute the objective function value f_z of the combination z, where $z = 1, \ldots, M$.

Step 6) Compute the main effect S_{jk}, where $j = 1, \ldots, N$, and $k = 1, 2$.

Step 7) Determine the better one of two levels of each factor based on the main effect.

Step 8) The next move $\mathbf{X}_i(t+1)$ is formed using the combination of the better partial vectors.

Step 9) Verify that $\mathbf{X}_i(t+1)$ is superior to $\mathbf{X}_i(t)$ and the M sampling solutions derived from the OA combinations. If it is not true, let $\mathbf{X}_i(t+1)$ be the best one among $\mathbf{X}_i(t)$ and these M sampling solutions.

The OED has been proven optimal for additive and quadratic models, and the selected combinations are good representations for all of the possible combinations [12]. The OA specifies a small number M of representative combinations that are uniformly distributed over the neighborhood of the position $\mathbf{X}_i(t)$. The number of objective function evaluations is $M+1$ per IMM operation, which includes M evaluations in Step 5) and one in Step 9). The IMM spends $M+1$ function evaluations, whereas the move mechanism of PSO spends one function evaluation using the generate-and-go method to determine the next move. However, the M sampling solutions and the factor analysis make the IMM more efficient in obtaining the next move. If interactions among partial vectors are weak, $\mathbf{X}_i(t+1)$ is a potentially good approximation to the best one of all the 2^N combinations in the neighborhood of $\mathbf{X}_i(t)$.

C. OPSO Algorithm

The simple OPSO algorithm with IMM for the optimization function f is described here. All efficient strategies of control-parameter specification and heuristics which are good for PSO may be also good for OPSO.

Step 1) Randomly initialize $\mathbf{X}_i(t)$ and $\mathbf{V}_i(t)$, and compute $f(\mathbf{X}_i)$, where $i = 1, \ldots, \mathrm{Np}$, and $t = 0$.

Step 2) Compute $\mathbf{P}_i(t)$ for each particle and $\mathbf{P}_g(t)$ of the swarm.

Step 3) Apply the IMM to generate $\mathbf{X}_i(t+1)$ for each $\mathbf{X}_i(t)$, $i = 1, \ldots, \mathrm{Np}$.

Step 4) $V_i(t+1) = \mathbf{X}_i(t+1) - \mathbf{X}_i(t)$, $i = 1, \ldots, \mathrm{Np}$.

Step 5) If a prespecified stopping condition is met, stop the algorithm; otherwise, increase the value of t by one and go to Step 2).

The overhead of IMM in preparing the OA experiments and the factor analysis is relatively small compared with the cost of function evaluations. Let G be the number of total iterations. The number of IMM operations is $\mathrm{Np} \cdot G$. The time complexity of OPSO is $\mathrm{Np} + \mathrm{Np} \cdot G(M+1)$ function evaluations, where $M+1$ is the number of function evaluations per IMM operation. The complexity of PSO is $\mathrm{Np} + \mathrm{Np} \cdot G$

TABLE I
PERFORMANCE COMPARISONS OF VARIOUS ALGORITHMS FOR A PARAMETRIC FUNCTION WITH A GLOBAL OPTIMUM OF 121.598

	SGA	Stochastic GA	Sensitivity GA	GA-Local	SAGA	AGA	OEGA	PSO	OPSO
Chromosome length	1000 (Bit)	500 (Bit)	300 (Bit)	300 (Bit)	500 (Bit)	500 (Bit)	500 (Bit)	Real code	Real code
Population Size	51	31	51	51	30	30	30	20	5
Generation	750	200	NA	400	199	232	196	600	37
N_{eval}	38,250	12,000	11,259	37,493	12,000	12,000	12,000	12,000	12,000
Function value	69	115	114	105	88.114	45.816	79.800	113.705	116.183

function evaluations. When the OPSO is compared with the PSO using the same number of function evaluations, the OPSO uses a smaller number of iterations.

Generally, research works limited the value of Np to the range of 20–60 [6]. It is suggested that even though there is a slight improvement to the optimal solution with increasing swarm size, it increases the number of function evaluations to converge to an error limit. Since the IMM can efficiently exploit the neighborhood of a particle, the OPSO generally uses a relatively small value of Np.

IV. PERFORMANCE COMPARISONS OF OPSO

Kennedy [27] indicated that the local version of PSO might perform better on complex problems, whereas the global version of PSO would perform better on simple problems. Mendes *et al.* [26] indicated that one cannot find an efficient and general topology structure for any kinds of functions. Similarly, there are two versions of OPSO. To avoid from involving the specification of neighborhood topology and evaluate its generalization ability, the global version of OPSO with constants $w = 0.9$ and $c_1 = c_2 = 2$ is used for all performance comparisons in this paper. Let the optimization functions have D parameters. The used OA is $L_{N+1}(2^N)$, where $N = 2^{\lfloor \log_2(D+1) \rfloor} - 1$.

Four versions of PSO using benchmark functions (see Table II) are evaluated: global versus local version (with a neighborhood size of five) and inertia weight ($w = 0.9$ and $c_1 = c_2 = 2$) versus constriction factor ($w = 0.72984$ and $c_1 = c_2 = 1.4962$). The best version of PSO is selected for comparisons with OPSO, which uses the local version with a constriction factor. In this section, the PSO uses $\mathrm{Np} = 20$, and the OPSO uses $\mathrm{Np} = 5$. Both the OPSO and the PSO do not use any local search or heuristics which can improve their performances for specific problems. For fair comparisons, the OPSO and the PSO use the same prespecified number of function evaluations (N_{eval}) as the stopping condition.

A. Large Parameter Optimization Problem

KrishnaKumar *et al.* [28] proposed three approaches, namely, stochastic GA, sensitivity GA, and GA-local search, and provided reasonable success on the large parameter optimization problems using the test function $f(\mathbf{X})$ with $D = 100$ as follows:

$$\max \quad f(\mathbf{X}) = -\sum_{i=1}^{D} \left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right) \right] \tag{9}$$

where $\mathbf{X} = [x_1, \ldots, x_D]^{\mathrm{T}}$, and variable $x_i \in [3, 13]$. The performances of the aforementioned three methods and a simple GA (SGA) are obtained from [28]. To demonstrate a high performance of OPSO, the OPSO is additionally compared with the following popular GAs: SAGA [29], GA with adaptive probabilities of crossover and mutation (AGA) [30], and SGA with an elitist strategy using one-point crossover (OEGA) [31]. The results of SAGA, AGA, and OEGA are obtained from [22]. Since the performance of the stochastic GA is very good, both the PSO and the OPSO use $N_{\text{eval}} = 12\,000$ for a direct comparison. The average performances of all compared algorithms using ten independent runs are shown in Table I. The results reveal that the OPSO can efficiently obtain the best solution 116.183 where the solution of PSO is 113.705.

B. Parametric Optimization Function

To evaluate the efficiency of OPSO for solving optimization functions with various dimensions, 12 benchmark functions gleaned from the literature, including unimodal and multimodal functions, continuous and discontinuous functions, smooth and nonsmooth functions, and function with and without interactions among parameters, are tested in the experimental study. The test function, the variable domain, and the global optimum for each function with D parameters are obtained from [22] and listed in Table II.

The average, best, and worst performances of PSO and OPSO using 30 independent runs for $D = 10$ and 100 are listed in Tables III and IV. A paired Wilcoxon test is applied to the results of PSO and OPSO, and the p values are also given. The experimental results reveal that the OPSO performs well and is significantly better than the PSO for most functions in a limited amount of computation time.

In order to show that the proposed IMM is effective, the simple PSO and OPSO without heuristics are further compared with the following GAs with elitist strategy and the associated commonly used crossover, which are denoted as

TABLE II
BENCHMARK FUNCTIONS

Test function	x_i domain	Optimum
$f_1 = -\sum_{i=1}^{D}\left[\sin(x_i)+\sin\left(\frac{2x_i}{3}\right)\right]$	[3, 13]	1.21598D(max)
$f_2 = -\sum_{i=1}^{D-1}\left[\sin(x_i+x_{i+1})+\sin\left(\frac{2x_ix_{i+1}}{3}\right)\right]$	[3,13]	≈2D(max)
$f_3 = \sum_{i=1}^{D}\lfloor x_i+0.5\rfloor^2$	[-100, 100]	0(min)
$f_4 = \sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$	[-5.12, 5.12]	0(min)
$f_5 = \sum_{i=1}^{D} x_i^2$	[-5.12, 5.12]	0(min)
$f_6 = \sum_{i=1}^{D}(x_i\sin(10\pi x_i))$	[-1.0, 2.0]	1.85D(max)
$f_7 = \sum_{i=1}^{D}\left\lvert\frac{\sin(10x_i\pi)}{10x_i\pi}\right\rvert$	[-0.5, 0.5]	0(min)
$f_8 = 20+e-20e^{-0.2\sqrt{\frac{\sum_{i=1}^{D}x_i^2}{D}}}-e^{\frac{\sum_{i=1}^{D}\cos(2\pi x_i)}{D}}$	[-30, 30]	0(min)
$f_9 = 418.9829D-\sum_{i=1}^{D}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	[-500, 500]	0(min)
$f_{10} = \sum_{i=1}^{D-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	[-5.12, 5.12]	0(min)
$f_{11} = 6D+\sum_{i=1}^{D}\lfloor x_i\rfloor$	[-5.12, 5.12]	0(min)
$f_{12} = \frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]	0(min)

TABLE III
FUNCTION VALUE COMPARISONS FOR $D = 10$

	PSO			OPSO			Wilcoxon test
	Average	Best	Worst	Average	Best	Worst	p value
f_1	11.8620	12.1598	11.1671	12.1598	12.1598	12.1598	0.0459
f_2	13.7289	17.1099	9.3706	17.1258	17.3182	15.3939	0.0
f_3	0	0	0	0	0	0	1.0
f_4	10.1588	6.9717	14.9394	2.2517	2.0057	2.8272	0.0
f_5	0	0	0	0	0	0	1.0
f_6	15.8619	17.9119	13.9965	17.2476	18.3028	16.1915	0.0
f_7	0.0107	0.0087	0.0125	0.0087	0.0087	0.0087	0.0078
f_8	0.2586	0.2586	0.2586	0	0	0	0.0
f_9	3745.7	3633.0	3870.6	0.1011	0.0593	0.3160	0.0
f_{10}	5.3672	4.1851	8.2574	0.3903	0.3406	0.5891	0.0
f_{11}	8.80	8.80	8.80	8.80	8.80	8.80	1.0
f_{12}	1.000	1.000	1.000	1.000	1.000	1.000	1.0

TABLE IV
FUNCTION VALUE COMPARISONS FOR $D = 100$

	PSO			OPSO			Wilcoxon test
	Average	Best	Worst	Average	Best	Worst	p value
f_1	109.3581	113.1074	104.0918	110.6192	115.0793	105.1583	0.1103
f_2	98.6653	118.0126	77.6472	107.2632	112.2602	94.4153	0.0333
f_3	16.2445	6.9319	31.4023	6.2320	2.0716	9.0056	0.0005
f_4	491.8799	454.2761	543.5168	431.3616	378.7336	483.3120	0.0
f_5	20.4402	3.2229	58.5398	2.4391	2.3286	3.2549	0.0
f_6	84.6553	103.9308	72.5705	87.6406	99.0162	79.5168	0.5946
f_7	0.1133	0.1025	0.1237	0.1130	0.1126	0.1138	0.6898
f_8	3.3754	2.9934	4.7839	1.7354	1.5570	1.9267	0.0
f_9	38028	36543	38709	14421	12544	16959	0.0
f_{10}	1864	1105	3691	8098	4616	11024	0.0
f_{11}	361.40	303.04	466.88	104.17	95.15	110.18	0.0
f_{12}	14.00	7.98	25.70	7.45	4.68	10.79	0.0

OEGA (one-point), TEGA (two-point), UEGA (uniform), and BLXGA (BLX-α) [32]. Each parameter is encoded using 10 bit for all test functions except that functions f_9 and f_{12} use 24 bit. The dimensions of the 12 test functions vary on $D = 10, 20, \ldots, 100$ in the experiments. The stopping condition uses $N_{\text{eval}} = 10\,000$ for all compared algorithms.

To illustrate the performance comparisons on various numbers of parameters, we use a distance function $\text{Dist}(D)$ for describing the mean distance between the optimal solution $f_{\text{opt}}(D)$ and the obtained best solution $f_{\text{best}}(D)$ for one parameter as follows:

$$\text{Dist}(D) = |f_{\text{opt}}(D) - f_{\text{best}}(D)|/D. \tag{10}$$

The results of average $\text{Dist}(D)$ for all test functions with $D = 10, 20, \ldots, 100$ using 30 independent runs are shown in Fig. 1 and Tables V and VI. The results of all GAs are obtained from [22]. Fig. 1 shows that the mean-distance values $\text{Dist}(D)$

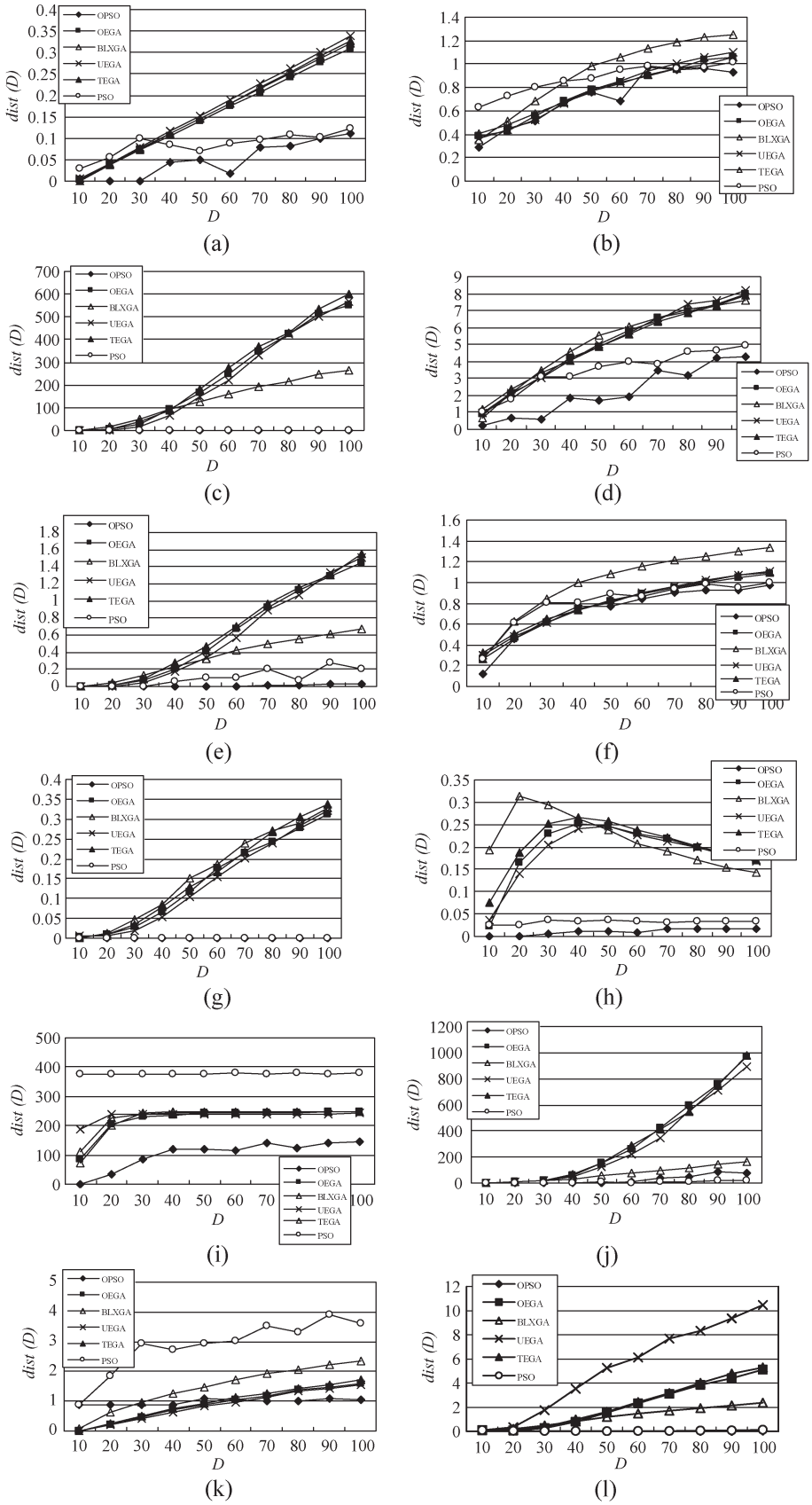

Fig. 1. Comparisons of various algorithms using $\mathrm{Dist}(D)$ curves for functions $f_1, f_2, \ldots$, and f_{12} in (a), (b), . . ., and (l), respectively.

TABLE V
FUNCTION VALUES AND RANKS FOR FUNCTIONS WITH $D = 10$

Function	OEGA	UEGA	TEGA	BLXGA	PSO	OPSO
f_1	12.119(3)	12.110(5)	12.113(4)	12.150(2)	11.862(6)	12.159(1)
f_2	16.19(4)	16.39(3)	15.89(5)	16.52(2)	13.72(6)	17.12(1)
f_3	5.53(5)	5.10(4)	4.90(3)	5.83(6)	0(1)	0(1)
f_4	8.94(3)	9.57(4)	11.63(6)	6.48(2)	10.15(5)	2.25(1)
f_5	0.0004(4)	0.0003(3)	0.0006(5)	0.0092(6)	0(1)	0(1)
f_6	15.82(4)	15.54(5)	15.24(6)	15.85(3)	15.86(2)	17.24(1)
f_7	0.035(4)	0.040(6)	0.039(5)	0.017(3)	0.010(2)	0.008(1)
f_8	0.23(2)	0.36(4)	0.75(5)	1.93(6)	0.25(3)	0(1)
f_9	848.1(3)	1859.5(5)	1132.0(4)	714.7(2)	3745.7(6)	0.1(1)
f_{10}	41.39(6)	39.56(5)	23.00(4)	22.63(3)	5.36(2)	0.39(1)
f_{11}	0.030(1)	0.067(2)	0.170(3)	1.030(4)	8.800(5)	8.800(5)
f_{12}	1.001(3)	1.030(6)	1.002(4)	1.030(5)	1.000(1)	1.000(1)
Avg rank	3.5	4.33	4.5	3.66	3.33	1.33
Final rank	3	5	6	4	2	1

TABLE VI
FUNCTION VALUES AND RANKS FOR FUNCTIONS WITH $D = 100$

Function	OEGA	UEGA	TEGA	BLXGA	PSO	OPSO
f_1	90.52(3)	89.36(4)	87.86(6)	88.9(5)	109.35(2)	110.61(1)
f_2	94.59(3)	89.49(5)	94.27(4)	74.47(6)	98.66(2)	107.26(1)
f_3	55027(4)	56647(5)	60252(6)	26320(3)	16.24(2)	6.23(1)
f_4	795.75(5)	819.65(6)	791.56(4)	763.42(3)	491.87(2)	431.36(1)
f_5	145.52(4)	151.43(5)	154.97(6)	67.89(3)	20.44(2)	2.43(1)
f_6	76.72(3)	74.50(5)	76.06(4)	51.53(6)	84.65(2)	87.64(1)
f_7	3.86(4)	3.94(5)	3.82(3)	5.45(6)	0.1133(2)	0.1130(1)
f_8	16.93(5)	16.83(4)	16.97(6)	14.35(3)	3.37(2)	1.73(1)
f_9	24645(3)	24160(2)	24723(4)	24794(5)	38028(6)	14421(1)
f_{10}	96556(5)	89514(4)	97990(6)	16086(3)	1864(1)	8098(2)
f_{11}	159.90(3)	154.37(2)	172.03(4)	233.63(5)	361.40(6)	104.17(1)
f_{12}	511.93(4)	1002.00(6)	537.09(5)	237.84(3)	14.00(2)	7.45(1)
Avg rank	3.83	4.41	4.83	4.25	2.58	1.08
Final rank	3	5	6	4	2	1

of OPSO and PSO slightly increase while D increases from 10 to 100 compared with other algorithms. Tables V and VI reveal that the OPSO and the PSO have the best and second performances with final ranks 1 and 2, respectively. From these experimental results, it reveals that the OPSO performs well in efficiently solving small and large parameter optimization problems in a limited amount of computation time.

Recently, two OED-based algorithms, namely, IEA [22] and OSA [24], which are efficient for solving the large parameter optimization problems were proposed. All the three OED-based algorithms have the same superiority for the large parameter optimization problems. The common scenario is that the simple IEA, OSA, and OPSO can significantly improve the conventional GAs, SA, and PSO, respectively, for large parameter optimization problems. Due to different aims and merits, the comparisons among the three OED-based algorithms are not discussed in this paper.

V. TASK ASSIGNMENT PROBLEMS

A. Problem Description [8]

Task assignment is one of the core steps to effectively exploit the capabilities of distributed or parallel computing systems.

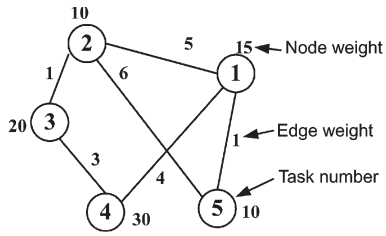

Fig. 2. TIG instance on a homogeneous system.

The task assignment problem is that of assigning tasks of a program among different processors of a distributed computer system in order to reduce the program turnaround time and to increase the system throughput. A distributed program is characterized by task interaction graph (TIG) $G(V_{\rm T}, E_{\rm T})$, where $V_{\rm T} = \{1, \ldots, m\}$ represents the tasks of a program, and $E_{\rm T}$ models the interaction between these tasks. The edge weight e_{ij} between nodes i and j denotes the information exchange between these pair of tasks. The node weight w_{ij} corresponds to the work to be performed by task i on processor j. In a homogeneous system, processors have the same computing power, and each task takes the same amount of time to execute on each processor. Fig. 2 shows an instance of the problem on a homogenous system.

The underling computing system is represented by a processor interaction graph (PIG) ($V_{\rm P}$, $E_{\rm P}$), where $V_{\rm P} = \{1, \ldots, n\}$ represents the set of processors in the system, and $E_{\rm P}$ represents the set of communication links between these processors. An edge weight d_{kj} between any two nodes k and j represents the length of the shortest path between the corresponding processors. The assignment problem can be formally represented as follows:

$$A : V_{\rm T} \rightarrow V_{\rm P} \tag{11}$$

where A is a function that maps the set of tasks $V_{\rm T}$ to the set of processors $V_{\rm P}$. $A(i) = j$ if task i is assigned to processor j. Let m be the total number of tasks and n be the total number of processors, where $1 \leq i \leq m$, $1 \leq j \leq n$, and then, the objective function (a minimax cost model) is described next.

Find a mapping instance A, such that when estimating the total time required by each processor, the largest time among all processors is minimized. Let $C_{\rm exe}(A)_k$ be the total execution time of all tasks assigned to processor k, such that

$$C_{\rm exe}(A)_k = \sum_i w_{ik} \qquad \forall A(i) = k \tag{12}$$

and let $C_{\rm com}(A)_k$ be the total interaction time between the tasks assigned to processor k and those that are not assigned to that processor in an assignment A, i.e.,

$$C_{\rm com}(A)_k = \sum_i \sum_j d_{A(i)A(j)} e_{ij} \qquad \forall A(i) = k, \quad \text{and} \quad A(j) \neq k. \tag{13}$$

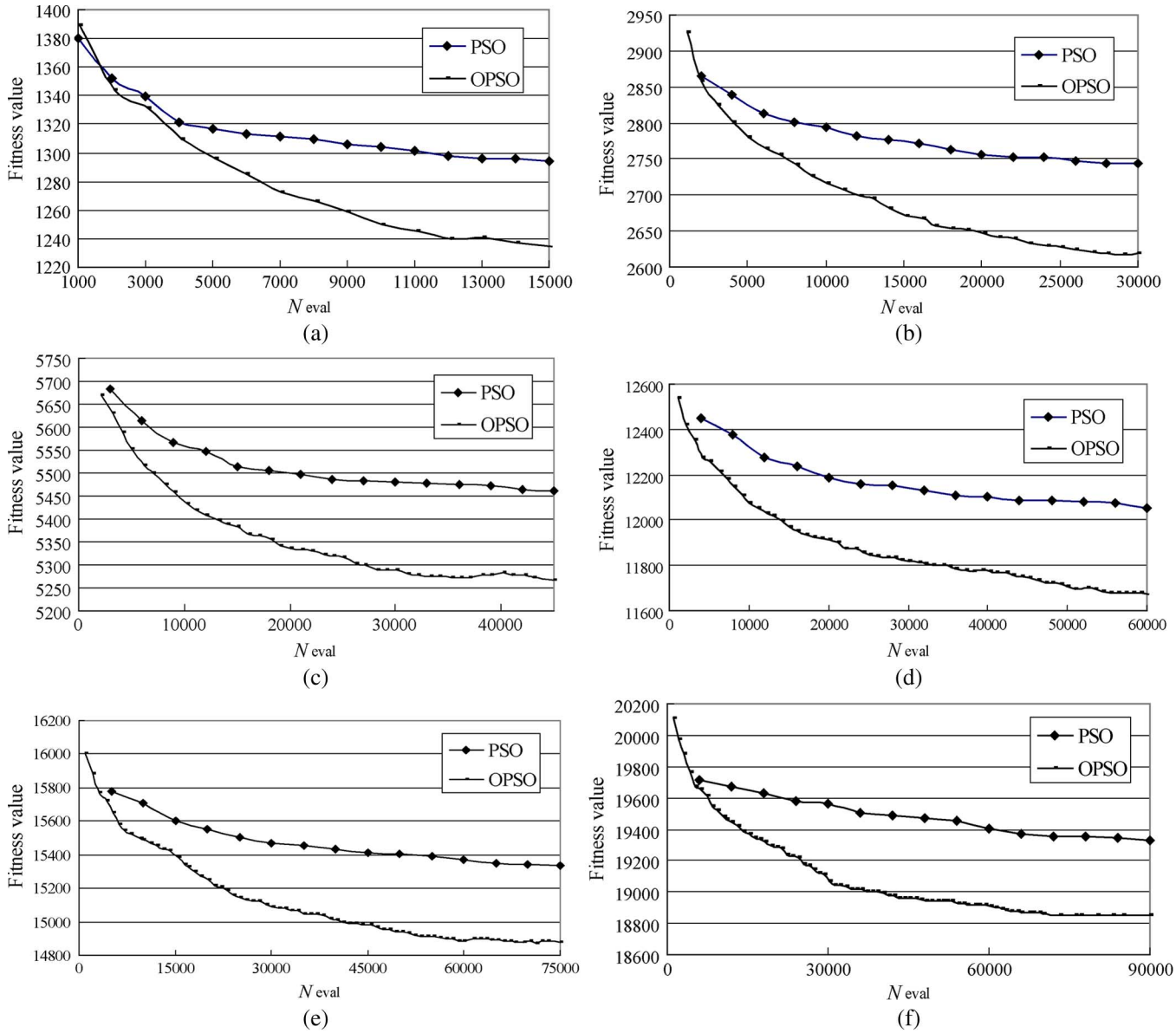

Fig. 3. Performance comparisons of PSO and OPSO for various numbers m of tasks. (a) $m = 50$. (b) $m = 100$. (c) $m = 150$. (d) $m = 200$. (e) $m = 250$. (f) $m = 300$.

Then, for a given assignment A, $C_{\text{total}}(A)_k$, which is the total time of a processor k, will be equal to the sum of the execution time and the interaction time for processor k, i.e.,

$$C_{\text{total}}(A)_k = C_{\text{exe}}(A)_k + C_{\text{com}}(A)_k. \tag{14}$$

Then, the final cost of an assignment will be dominated by the processor with the highest $C_{\text{total}}(A)_k$, i.e.,

$$\text{Cost}(A) = \max\left(C_{\text{total}}(A)_k \text{ for } 1 \le k \le n\right). \tag{15}$$

The goal of the task assignment problem is to find an assignment A which has the minimum cost for a given TIG on a given PIG, i.e.,

$$\text{minimize Cost}(A). \tag{16}$$

B. Performance Comparison

The source code of the compared PSO and data sets is obtained from Salman *et al.* [8]. A particle is represented as $\mathbf{X} = [x_1, \ldots, x_m]^{\mathrm{T}}$, where m is the number of tasks. Each variable x_i has a discrete set of possible values belonging to $\{1, \ldots, n\}$, where n is the number of processors in the distributed system under consideration. The particle representations (mapping between problem solution and particle) of PSO and OPSO are the same. The control parameters of OPSO are as follows: $\text{Np} = 30$, $w = 0.9$, and $c_1 = c_2 = 2.0$. If $m > 100$, the OA table uses $L_{16}(2^{15})$ with $N = 15$; otherwise, $L_8(2^7)$ with $N = 7$. The stopping condition uses the same number of N_{eval} with that of PSO using $\text{Np} = 2m$ and 150 generations [8].

The test data consist of six randomly generated TIGs. The size of TIG is varied from $m = 50$ to 300 nodes with an

TABLE VII
COST COMPARISONS FOR TASK ASSIGNMENT PROBLEMS WITH m TASKS

Method	PSO	OPSO	PSO	OPSO	PSO	OPSO	PSO	OPSO	PSO	OPSO	PSO	OPSO
m	50	50	100	100	150	150	200	200	250	250	300	300
Best	1228	1127	2690	2537	5237	5076	11823	11337	14861	14622	19132	18537
Worst	1357	1290	2812	2700	5539	5391	12293	11875	15549	15238	19599	19155
Mean	1295.1	1233.4	2748.8	2617.6	5458.2	5253.3	12044.5	11699.7	15308.6	14870.6	19388.2	18862.1
Variance	905.7	1793.2	757.9	1998.8	4752.6	6751.0	16862.8	20583.2	27833.8	24945.3	18215.1	25872.1

increment of 50. We compare the PSO with the OPSO using ten fully connected homogeneous processors. Therefore, there is no variable which is independent in the optimization problem. For each TIG, both the PSO and the OPSO conducted 30 independent runs. Fig. 3 shows the average convergences of PSO and OPSO for various numbers m of tasks. Table VII shows the cost results of PSO and OPSO.

The performance of the PSO in solving the task assignment problem is evaluated in comparison with the GA for a number of randomly generated mapping problem instances in [8]. The results showed that the solution quality of PSO is better than that of GA in most of the test cases. According to the results of convergence and cost value (Fig. 3 and Table VII), we can see that the OPSO is significantly better than the PSO in solving task assignment problems.

The task assignment problem with up to 300 tasks is a large parameter optimization problem which is NP-complete. Furthermore, there are strong interactions among the parameters of the task assignment problem. These interactions will result in the so-called linkage of GA. Linkage represents the ability of building blocks to bind tightly together. For the problem with strong interactions, the conventional crossover operator of GA usually disrupts building blocks and, thus, reduces the GA performance. The move mechanism can probe a wider region of search space than the OEGA. Furthermore, the probed space of the move mechanism is smoother than that of the crossover of GA in the late evolution. It would increase the chance to find a better solution than the current best solution. As results, the PSO is better than the GA in solving the task assignment problem, and the OPSO using the divide-and-conquer strategy is better than the PSO in solving the large task assignment problem.

VI. CONCLUSION

This paper has proposed a novel variant of PSO, which is named orthogonal PSO. OPSO is more specialized than PSO and performs well on large-scale parameter optimization problems with few interactions between variables. In this paper, it is shown that the OPSO performs well in solving the real-valued parametric functions and the large discrete task assignment problem which is NP-complete in a limited amount of computation time. This is achieved by the proposed novel move behavior in the swarm: an IMM using a divide-and-conquer strategy. The OPSO-based method can be widely used to efficiently solve various applications of optimization problems. We believe that the problem-dependent learning strategies and auxiliary techniques can further advance the performance of OPSO. From our computer simulation results, the intelligent multiobjective version of OPSO is also efficient in solving multiobjective large-scale parametric optimization functions compared with the existing multiobjective versions of PSO [33].

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, Australia, Dec. 1995, vol. IV, pp. 1942–1948.
[2] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming*, vol. VII, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 591–600.
[3] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Conf. Evol. Comput.*, Anchorage, AK, 1998, pp. 69–73.
[4] J. Kennedy, "The behavior of particles," in *Evolutionary Programming*, vol. VII, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 581–590.
[5] K. E. Parsopoulos and M. N. Vrahatis, "Initializing the particle swarm optimizer using the nonlinear simplex method," in *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, A. Grmela and N. Mastorakis, Eds. Interlaken, Switzerland: WSEAS Press, 2002, pp. 216–221.
[6] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. IEEE Int. Conf. Evol. Comput.*, 2000, vol. 1, pp. 84–88.
[7] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
[8] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocess. Microsyst.*, vol. 26, no. 8, pp. 363–371, Nov. 2002.
[9] C.-C. Chiu, Y.-S. Yen, and J.-S. Chou, "A fast algorithm for reliability-oriented task assignment in a distributed system," *Comput. Commun.*, vol. 25, no. 17, pp. 1622–1630, Nov. 2002.
[10] J. W. Chinneck, V. Pureza, R. A. Goubran, G. Karam, and M. Lavoie, "A fast task-to-processor assignment heuristic for real-time multiprocessor DSP applications," *Comput. Oper. Res.*, vol. 30, no. 5, pp. 643–670, Apr. 2003.
[11] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.
[12] Q. Wu, "On the optimality of orthogonal experimental design," *Acta Math. Appl. Sin.*, vol. 1, no. 4, pp. 283–299, 1978.
[13] A. Dey, *Orthogonal Fractional Factorial Designs*. New York: Wiley, 1985.
[14] A. S. Hedayat, N. J. A. Sloane, and J. Stufken, *Orthogonal Arrays: Theory and Applications*. New York: Springer-Verlag, 1999.

[15] T.-P. Bagchi, *Taguchi Methods Explained: Practical Steps to Robust Design*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[16] J.-T. Tsai, T.-K. Liu, and J.-H. Chou, "Hybrid Taguchi-genetic algorithm for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 4, pp. 365–377, Aug. 2004.
[17] H. Tanaka, "A comparative study of GA and orthogonal experimental design," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1997, pp. 143–146.
[18] Q. Zhang and Y.-W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 53–62, Apr. 1999.
[19] Y.-W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 5, no. 1, pp. 41–53, Feb. 2001.
[20] S.-Y. Ho, H.-M. Chen, S.-J. Ho, and T.-K. Chen, "Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 1031–1044, Apr. 2004.
[21] H.-L. Huang and S.-Y. Ho, "Mesh optimization for surface approximation using an efficient coarse-to-fine evolutionary algorithm," *Pattern Recognit.*, vol. 36, no. 5, pp. 1065–1081, May 2003.
[22] S.-Y. Ho, L.-S. Shu, and J.-H. Chen, "Intelligent evolutionary algorithms for large parameter optimization problems," *IEEE Trans. Evol. Comput.*, vol. 8, no. 6, pp. 522–541, Dec. 2004.
[23] S.-Y. Ho, S.-J. Ho, Y.-K. Lin, and W. C.-C. Chu, "An orthogonal simulated annealing algorithm for large floorplanning problems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 8, pp. 874–877, Aug. 2004.
[24] S.-J. Ho, S.-Y. Ho, and L.-S. Shu, "OSA: Orthogonal simulated annealing algorithm and its application to designing mixed H_2/H_∞ optimal controllers," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 5, pp. 588–600, Sep. 2004.
[25] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
[26] B. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 204–210, Jun. 2004.
[27] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *Proc. Conf. Evol. Comput.*, Washington DC, 1999, pp. 1931–1938.
[28] K. KrishnaKumar, S. Narayanaswamy, and S. Garg, "Solving large parameter optimization problems using a genetic algorithm with stochastic coding," in *Genetic Algorithms in Engineering and Computer Science*, G. Winter, J. Périaux, M. Galán, and P. Cuesta, Eds. Hoboken, NJ: Wiley, 1995, pp. 287–303.
[29] H. Esbensen and P. Mazumder, "SAGA: A unification of the genetic algorithm with simulated annealing and its application to macro-cell placement," in *Proc. IEEE Int. Conf. VLSI Des.*, 1994, pp. 211–214.
[30] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 4, pp. 656–667, Apr. 1994.
[31] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[32] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval schemata," in *Foundation of Genetic Algorithm-2*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993.
[33] S.-J. Ho, W.-Y. Ku, J.-W. Jou, M.-H. Hung, and S.-Y. Ho, *Intelligent Particle Swarm Optimization in Multi-Objective Problems*, vol. 3918. Berlin, Germany: Springer-Verlag, 2006, pp. 790–800.

Shinn-Ying Ho (M'00) was born in Taiwan, R.O.C., in 1962. He received the B.S., M.S., and Ph.D. degrees in computer science and information engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1984, 1986, and 1992, respectively.

From 1992 to 2004, he was with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan. He is currently a Professor with the Department of Biological Science and Technology and the Institute of Bioinformatics, National Chiao Tung University. His research interests include evolutionary algorithms, soft computing, image processing, pattern recognition, bioinformatics, data mining, machine learning, computer vision, fuzzy classifier, large-scale parameter optimization problems, and system optimization.

Hung-Sui Lin was born in Taiwan, R.O.C., in 1977. He received the B.S. degree in information management from the National Defense of Management College, Taipei, Taiwan, in 1999, and the M.S. degree in information engineering and computer science from Feng Chia University, Taichung, Taiwan, in 2004.

He is currently an Officer with the Office of the Deputy Chief of the General Staff for Communications, Electronics and Information, Ministry of National Defense, Taiwan. His research interests include evolutionary algorithms, large parameter optimization problems, system optimization, software development, and information security.

Weei-Hurng Liauh was born in Taiwan, R.O.C., in 1960. He received the B.S. degree in computer engineering from the Chung Cheng Institute of Technology, Taoyuan, Taiwan, in 1987, and the M.S. degree from Feng Chia University, Taichung, Taiwan, in 2004.

His research interests include evolutionary computation and system optimization.

Shinn-Jang Ho was born in Taiwan, R.O.C., in 1960. He received the B.S. degree in power mechanic engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1983, and the M.S. and Ph.D. degrees in mechanical engineering from the National Sun Yat-Sen University, Kaohsiung, Taiwan, in 1985 and 1992, respectively.

He is currently a Professor with the Department of Automation Engineering, National Formosa University, Yunlin, Taiwan. His research interests include optimal control, fuzzy systems, evolutionary algorithms, and system optimization.