IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 9, 2017, accepted March 22, 2017, date of publication April 17, 2017, date of current version June 7, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2694843

Query-Based Learning for Dynamic Particle Swarm Optimization

RAY-I CHANG[1], (Member, IEEE), HUNG-MIN HSU[1,2], SHU-YU LIN[1,3], CHU-CHUN CHANG[1], AND JAN-MING HO[4], (Senior Member, IEEE)

[1]National Taiwan University, Taipei 10617, Taiwan
[2]Research Center for Information Technology Innovation, Academia Sinica, Taipei 115, Taiwan
[3]National Chung-Shan Institute of Science and Technology, Taoyuan 325, Taiwan
[4]Institute of Information Science and Research Center for Information Technology Innovation, Academia Sinica, Taipei 115, Taiwan

Corresponding author: Hung-Min Hsu (tsbear1@gmail.com)

This work was supported in part by the Academia Sinica Digital Center: System Management and Content Retrieval Technologies for supporting Cloud-based Digital Archive Systems and Services Projects through the Academia Sinica and the Ministry of Science and Technology, Taiwan, under Grant 105-2410-H-002-099-MY3.

ABSTRACT In recent years, many researchers have examined dynamic optimization problems (DOPs). The key challenge lies in the fact that the optimal solution of a DOP typically changes over time. This paper focuses on using query-based learning dynamic particle swarm optimization (QBLDPSO) to solve DOPs. QBLDPSO is mainly used for improving multi-population-based PSO; our QBL mechanism includes two learning strategies that integrate the concepts of diversity and memory into PSO. The first learning strategy, QBL quantum parameter adaptation (QBLQPA), is used to apply the concept of diversity to the multi-population based algorithm. This is different from typical diversity-based PSO approaches, which passively maintain the diversity of particles in the solution space. We actively adapt the ratio of quantum particles and neutral particles to achieve diversity without analyzing the distribution of optima in the solution space. The second learning strategy is query-based learning optima prediction (QBLOP). Although QBLOP exploits the concept of memory, we do not need to analyze the history of all particles. We select the k nearest particles to the current best solution and use a minimum encompassing circle as the possible prediction region. Our experimental results are based on the generalized dynamic benchmark generator (GDBG), which is adopted as a benchmark for the DOP. The proposed method outperforms two state-of-the-art multi-population-based PSO methods with the average improvements of 11.37% and 8% using QBLQPA. In particular, for the recurrent problems in GDBG, our method improves performance by 35.06%.

INDEX TERMS Particle swarm optimization, dynamic optimization, quantum parameter adaptation, optima prediction, query-based learning.

I. INTRODUCTION

Recently, many researches have focused on stationary optimization problems. Thus, optimization algorithms for these problems have become robust and efficient. However, many real applications have optimal solutions that change over time. Optimization problems in these dynamic environments have become a challenging and vital issue. The fitness or constraint functions change over time when optimization problems occur in dynamic environments. Dynamic optimization problems (DOPs) cannot be solved efficiently by simply re-initializing the population. As a result, DOPs not only search for a global optimum, but also track the movement of optima.

Most researchers design a particular model for a specific environment because it is difficult to improve the performance of various DOPs using a general model. DOPs are categorized according to spatial and temporal severity [1]. In the first type of DOP, spatial severity is low and temporal severity is high. Consider step changes as an example, where the optimal solution for step changes shifts continuously. The second type of DOP is a recurrent problem, where optima emerge repeatedly or periodically. In the third type of DOP, the movement of the optimal solution is random or chaotic.

The problem in a DOP is to find the best solution and track its movement. There are three categories of PSO algorithms for DOPs: diversity, memory, and multi-population. We focus on improving multi-population based PSO. We propose a query-based learning dynamic particle swarm optimization (QBLDPSO) method to handle various

dynamic characteristics. QBL is a machine learning concept that uses a teacher, called an oracle, to guide learning behavior [2], [3]. PSO in QBLDPSO is referred to as a student. When PSO gets stuck in a local minimum, the oracle will provide guidance to redirect the particles. Based on the concept of QBL, two learning strategies for the oracle are introduced: query-based learning quantum parameter adaptation (QBLQPA) and query-based learning optima prediction (QBLOP). We demonstrate that controlling the ratio of neutral and quantum particles through QBLQPA is helpful for establishing better subpopulations to improve the PSO algorithm. Furthermore, we propose a QBLOP that finds the k nearest neighbor particles based on memory, and establishes a minimum encompassing circle that allows the predicted data to effectively improve the performance of multi-population-based PSO. The two QBL strategies first evaluate the environment (dynamic characteristics), then respond with suggestions for the PSO to continue searching while adaptively tuning parameters or redirecting particles toward the prediction.

This paper is organized as follows: the next section provides background information on DOPs, dynamic optimization benchmarks, and related approaches for DOPs. QBLDPSO is described in Section III, including QBLQPA and QBLOP. The proposed algorithm is experimentally compared to multi-population based PSO algorithms using the GDBG benchmark generator in Section IV. Section V contains our conclusions.

II. LITERATURE REVIEW

In this section, we discuss DOPs, benchmarks, and recent work in dynamic environments.

A. DOPs

There are many real-world problems that change over time, such as the dynamic knapsack problem, dynamic traveling salesman problem, and network routing problem [4]–[6]. Nguyen *et al.* [5] defined DOPs as follows. Given a problem F, if the fitness landscape of the problem F changes in the time period between $T1$ and $T2$, the problem F is referred as a DOP between $T1$ and $T2$.

The main challenge in DOPs is tracking moving optima. One important task for DOP algorithms is to detect when the environment changes. The algorithms then begin to track the moving optima. Most methods detect new changes, and otherwise consider the environment to be known. References [5] and [6] show that most benchmarks in DOPs are detectable, so changes can be detected using various agents. There are two main mechanisms used to detect changes. One is re-evaluating solutions, and the other is observing algorithm behavior. Re-evaluating solutions employs detectors to evaluate functions iteratively. If these detectors observe that optima are different from the previous iteration, it means that the environment is changed. For observing algorithm behavior, we can detect environment changes when fitness drops.

TABLE 1. Challenges and strategies in DOPs.

Challenges	Strategies
Change detection	Multi-agents used to detect changes or observe fitness curves.
Outdated memory	Re-evaluate particle positions and restore best particle positions
Diversity loss	Maintaining diversity in each iteration, increasing diversity when detecting environment changes, memory based approach, multi-population based approach
Recurrent problems	Use of memory concept to construct prediction model

However, two important issues are associated with PSO for tracking moving optima: outdated memory and diversity loss. The changing environment leads to outdated memory, meaning that previous experience is insufficient for PSO to find a global optimum or local optima. Diversity loss stems from PSO converging on previous global or local optimum. The solution for outdated memory is re-evaluating particle positions and restoring their best positions when the environment changes. For diversity loss, many optimization algorithms tackle the problem using various adaptations, such as maintaining diversity in each iteration, increasing diversity when detecting environment changes, memory based approaches, and multi-population based approaches.

In SOPs, changes occur during a short period of time and the optimal solution or peak moves in a smooth manner. On the other hand, many DOPs have regular patterns. The optimal solutions for these DOPs may re-appear after several environmental changes, these are called recurrent problems [1], [5]. Table 1 presents the challenges of DOPs and their solving strategies.

B. DOP BENCHMARK

We introduce the moving peak benchmark (MPB) and generalized dynamic benchmark generator (GDBG) in this section. MPB and GDBG were developed to enable comparison of different approaches for DOPs.

MPB is a famous benchmark in dynamic environments [7], [8]. MPB contains several peaks in its landscape. Each peak has its own characteristics, such as height, width, and position. The characteristics of each peak change slightly as the environment changes. Height is the highest point on each peak. Width is the drop rate of a point with respect to the distance from the center of a peak. The global best value of MPB is the maximum height of each peak, and the optimization objective of MPB is to search for the highest point in the landscape.

Similar to MPB, GDBG consists of several peaks, where the heights, widths, and positions of the peaks change over time [9]–[11]. The changes can be controlled using

system parameters. There are several types of system parameters, such as small step change, large step change, random change, chaotic change, recurrent change, and noisy recurrent change.

C. RECENT WORK IN DYNAMIC ENVIRONMENTS

Recently, several researches have attempted to adapt PSO algorithms for DOPs [1], [12]–[21]. Their approaches can be divided into three categories: diversity, memory, and multi-population.

The first category is the diversity approach. One of the key issues in DOPs is the convergence capabilities of the PSO algorithm. If PSO converges too fast, all particles converge to a single point in the search space. This means particles cannot find the optimum solution when the environment changes. As a result, these studies aim to prevent the phenomenon of premature convergence by introducing diversity after the environment changes or maintaining diversity during optimization. Introducing diversity requires detection of changes prior to the introduction. PSO will then increase diversity as the environment changes. Maintaining diversity means maintaining particles during the entire search process so that PSO does not have to detect changes. For this reason, PSO is also unable to track the moving optimum [12], [13], [15], [19], [21], [22].

The second category is the memory approach. Memory approaches try to derive patterns from previous search experiences to predict changes [1], [23], [24] or introduce new patterns to design self-adaptive mechanisms [25], [26].

The third category is the multi-population approach. Individuals are clustered into several sub-populations. Each sub-population is assigned to handle a different sub-region of the solution space. Additionally, the task for each sub-population can be different. Some sub-populations aim to track changes while others search for new optima [13], [15], [19], [27]. Thus, the multi-population approach must assign the tasks for each sub-population.

Blackwell et al. proposed a multi-swarm PSO [13], [15], [28] that incorporates two diversity approaches (charged PSO and quantum PSO) to explore several optima simultaneously. Multi-swarm PSO has two types of particles, neutral particles and charged or quantum particles. Neutral particles are used to update swarm optima rather than global optima. In contrast to the neutral particles, the charged or quantum particles try to preserve the diversity of swarms. In speciation-based PSO, Bird et al. dynamically regulated the number of neutral and quantum particles to enhance tracking ability in a dynamic environment [29].

On the other hand, Blackwell et al. proposed exclusion and anti-convergence concepts to maintain group diversity. The concept of exclusion is to avoid having different subgroups of particles focus on the same peak (local optima). If the distance between two groups of particles becomes too close, the exclusion strategy will activate to re-initialize the worst of the two subgroups to avoid both subgroups dropping into the same local optima. In terms of anti-convergence, the mechanism will be triggered to avoid local optima traps while the number of peaks is larger than the number of subgroups. Furthermore, both strategies will re-initialize the worst subgroup to find new optimum when all subgroups converge.

Multi-swarm PSO can be divided into two types: mQSO and mCPSO. mQSO uses quantum particles to seek optima, while mCPSO uses charged particles. Experimental results show that mQSO generally outperforms mCPSO. Thus, many approaches have been developed based on mQSO because mQSO is a powerful tool for solving DOPs. Novoa-Hernández et al. proposed *m*PSODE to improve mQSO. *m*PSODE is a combination of two strategies: a diversity strategy (mPSOD) and control rules (mPSOE) [19]. mPSOD uses quantum particles to cope with environment changes. mPSOD provides quantum particles when changes occur. These provided quantum particles are distributed around the global optimum. mPSOE determines whether certain swarms are in unpromising areas in the search space. mPSOE suspends these swarms using fuzzy rules to prevent numerous unnecessary computations.

We aim to combine PSO and QBL to handle optimization problems in dynamic environments because QBL has been proven to be a useful machine learning technique. Some previous works used the QBL technique to train a neural network based on a small amount of training data, and others solved various application based problems using QBL approaches [30], [31]. For example, QBL-PSO used QBL for swarm redistribution to enhance the exploitation and exploration abilities of PSO. According to QBL-PSO, PSO with swarm redistribution achieved superior convergence to a global optimum in the power contract problem [3]. MRPSO-QBL also used QBL to support PSO in order to escape local optima and aid in finding global optima while accelerating convergence speed [32].

III. PROPOSED METHOD

In this section, we first introduce the framework of QBLDPSO, and then further describe the two QBL strategies: QBLQPA and QBLOP.

A. QBLDPSO

QBLDPSO is a framework that provides a mechanism for PSO to interact with a learning process called QBL. QBLDPSO exploits the characteristics of DOPs to improve the performance of PSO.

PSO [33] simulates the social behavior of flocks of birds or schools of fish in order to solve complex problems. PSO optimizes a problem by using a population of particles. These particles are a population of candidate solutions that enable PSO to explore a large solution space. The movement of these particles is based on their own position and velocity. The velocity of a particle is driven by local best-known position of the particle and the best-known positions of other particles in the search space. PSO searches iteratively until a stop criterion is satisfied. During the search process, the

position and velocity of particle i is updated by the following equations:

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (1)$$

$$V_i(t+1) = \omega \times V_i(t) + c_1 r_1 \left(X_{pBest} - X_i(t)\right) + c_2 r_2 \left(X_{gBest} - X_i(t)\right) \quad (2)$$

where c_1 and c_2 are positive acceleration coefficients, and r_1 and r_2 are random numbers between zero and one. w is an inertial weight that is used to determine how previous velocity influences current velocity. *pBest* is the best position found so far by this particle and *gBest* is the best position found among all particles so far.

QBL is an active learning method. When a learner (PSO) is trapped in a difficult problem, the teacher (oracle) can provide useful information at the proper time to improve learning efficiency. Instead of using a fixed dataset to train the model as in passive learning methods, the strategy of QBL is to incrementally input training data during the learning progress. The process of QBL requires PSO to respond to problems, and the response of PSO is then given to the learning strategy model called the "oracle." The oracle will then guide PSO to search in promising areas. The complete learning process is called QBL due to the interaction with the query mechanism.

The role of the oracle in QBLDPSO is to detect problem characteristics. Because most DOPs change within specific restrictions and rules, the oracle attempts to analyze these characteristics to solve DOPs. Through QBL, we can retrieve the problem characteristics from the oracle. Initially, the QBLDPSO optimization process operates in in the same manner as the original PSO. However, the QBL model will record environment instances in an environment database. If the dynamic environment changes, the QBL model will analyze the environment characteristics and provide appropriate guidance to enhance the performance of PSO.

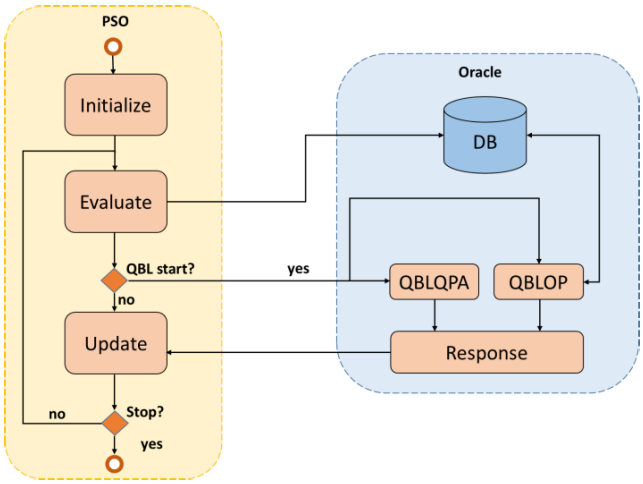


FIGURE 1. Framework of QBLDPSO.

Fig. 1 presents the framework of QBLDPSO. The left side shows the original PSO framework, and the oracle is shown on the right side. The oracle focuses on analyzing changes in the dynamic environment so that the QBL model can improve the tracking ability for moving optima. An "epoch" is the period between two environment changes. When an environment change is detected by the algorithm, the current epoch ends and a new epoch begins. The oracle learns problem characteristics by analyzing the best positions and fitness value within an epoch. Therefore, QBLDPSO uses the oracle to improve PSO when a new epoch begins. The oracle will provide guidance for PSO, and the influence of the oracle's guidance may continue for a brief duration (e.g., move partial particles to an assigned position a few generations after the change), or an enduring period (e.g., change the parameters of PSO for the remaining generations). The oracle employs two learning strategies (QBLQPA and QBLOP) to analyze environment characteristics. Thus, QBLDPSO can respond appropriately when the environment characteristics match the oracle's prediction.

B. ORACLE FOR QBLDPSO

There are two learning strategies employed by the oracle for QBLDPSO: QBLQPA and QBLOP. There were designed specifically to improve the PSO algorithm in this study. Their details are provided below.

One important issue in DOPs is maintaining diversity. In order to avoid premature convergence, we extend the multi-swarm, anti-convergence, and exclusion concepts in [15] to our algorithm, because these concepts have been proven to be a superior approach for DOPs [5]. In multi-swarm approaches, quantum particles are used to track the best particle positions. A quantum particle can efficiently track a moving optimal position by changing its movement rules. However, the movement rules for the quantum particle also decrease its exploitation abilities. Quantum particles are suitable for small changes in the environment. They do not provide good results for drastic changes in the environment. In other words, quantum particles reduce the efficiency of the algorithm.

We propose a quantum number self-adapting strategy called QBLQPA for our oracle, which controls the ratio between quantum and neutral particles based on the severity of changes in the environment. Quantum particles are suitable for solving a DOP where changes in the environment are gradual. In this case, QBLQPA raises the ratio of quantum to neutral particles. In contrast, if changes in the environment are abrupt, quantum particles are less beneficial, and QBLQPA will reduce the ratio of quantum to neutral particles in order to improve performance. The function for the quantum particle number decision is as follows:

$$Dist_t = |HK_t - HK_{t-1}| \quad (3)$$

$$P_{out} = Count\left\{Dist_t \mid Dist_t > W_R, t = 1, \ldots, T\right\}/T \quad (4)$$

$$N_Q = (1 - P_{out}) \times 0.9 \quad (5)$$

where HX_t denotes the best position in epoch t, $Dist_t$ is the distance from the best position in epoch $t$to the best position

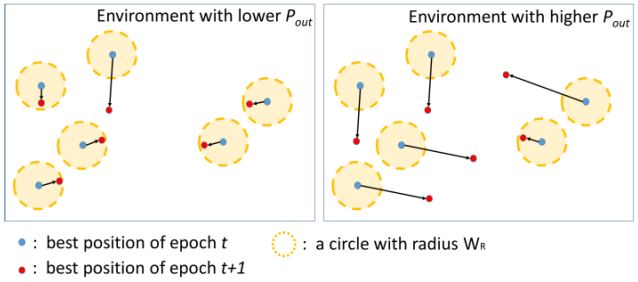


FIGURE 2. Concept of P_{out}.

in epoch $t-1$, W_R represents the effective radius of the quantum particles centered on the best position of the particle swarm, and T is the size of the environmental dataset being considered by the QBL model. P_{out} is the probability that the movement of the best position exceeds the effective range of the quantum particle. Furthermore, P_{out} also indicates the abruptness of change in the environment. N_Q is the ratio used to control the quantum particle number in QBLQPA. The concept of P_{out} in QBLQPA is shown in Fig. 2. The right side shows that QBLQPA detects an optimal solution for the next time unit outside the distance that the quantum particle can capture (large P_{out}). In this case, QBLQPA will automatically decrease the number of quantum particles (smaller N_Q). The left side depicts a different situation. If P_{out} is continuously low, QBLQPA will self-adaptively increase the number of quantum particles. However, our experiments indicate that we must reserve at least 10% of the particles as neutral to avoid QBLQPA transforming all particles into quantum particles, even in extreme environments. Thus, we use 0.9 to control the quantum particle number in equation (5). The default ratio of quantum particles to neutral particles is set to 0.5 based on the results of [15]. We set the basic requirement for QBLQPA activation to at least three instances in the environment database.

We establish QBLOP as the second strategy for our oracle. Some DOPs are predictable or partially predictable [5]. In QBLOP, we try to identify whether or not the problem is predictable. QBLOP will then guide PSO to seek the optimal solution. QBLOP aims to recognize certain characteristics in the current epoch via the database. Once a characteristic is identified, QBLOP will generate the prediction region with the highest probability of capturing the optimal solution in the next epoch.

QBLOP takes advantage of reference data from the database to predict the possible optimal location only when the reference data is considered reliable. Therefore, QBLOP defines two important parameters: reference severity and prediction severity.

Reference severity is used to validate the reliability of reference data. We denote the current search optima as HX_{now}. When changes are detected, QBLOP adopts the first k nearest data nodes to HX_{now} in the database as reference data. Reference severity is defined as the furthest distance between the reference data and HX_{now}.

$$D_R = \text{Max}\{|HX_i - HX_{now}|, \quad i = 1, \ldots, K\}. \tag{6}$$

When reference severity is high, the reliability of the reference data is low, which means there are no patterns in common with the previous situation. In contrast, low reference severity means that the reference data have patterns in common with HX_{now}. In this case, we can use the reference data to predict the location of the optimal solution.

Prediction severity is the metric used to assess the reliability of predicted data. After reference data are determined to be reliable, QBLOP will select the next search optimum of the reference data as the predicted data PX_t $(t = 1, \ldots, K)$. Then, a predicted possible region is dawn to cover the predicted data via a minimum encompassing circle. The definition of prediction severity is the diameter of the minimum encompassing circle:

$$D_p = M\text{ax}\{\left|PX_i - PX_j\right|, i = 1, \ldots, K, \\ j = 1, \ldots, K\} \times \sqrt{3}. \tag{7}$$

In terms of prediction severity, the predicted data is sparse and the minimum encompassing circle is large when prediction severity is high; in contrast, predicted data converge when prediction severity is low.

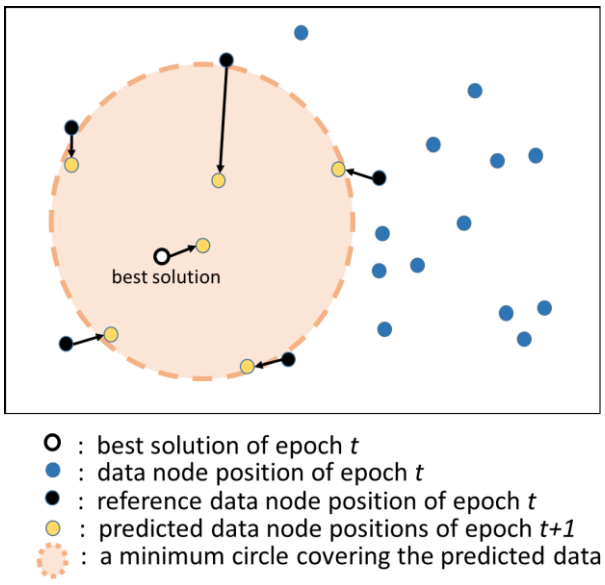


FIGURE 3. QBLOP.

QBLOP can be used to effectively cope with recurrent problems because the optima will reappear periodically. When the optima reappear, QBLOP can exploit the historical data of the same peak from the database and use the reference data to predict the next optimal solution. Fig. 3 shows the concept of QBLOP. QBLOP extracts the k nearest historical data from the best solution as reference data. QBLOP then reads the next position of the k reference data from database. Finally, QBLOP generates a minimum circle encompassing the next positions of these reference data and refers to the circle as a predicted possible region. The pink circle covering the predicted data indicates the possible region that optima may drop into during the next epoch. Eventually, QBLOP will

randomly generate neutral and quantum particles only inside this region.

In QBLOP, we have an important decision to make. That is deciding the thresholds for reference severity and prediction severity. In this study, we use the basin of attraction of a peak to determine the threshold for reference severity. DOPs typically have several peaks, and each peak has basin of attraction. Particles will be attracted to converge on a peak when they are distributed over the peak's basin of attraction. Blackwell and Branke [15] define the average diameter of the basin of attraction of a peak as d_{boa}. d_{boa} can be estimated by $d_{boa} = X/p^{1/d}$, where X is the dynamic range, p is the number of peaks, and d is the number of dimensions. Therefore, we use the concept of d_{boa} as the threshold for reference severity. If D_R becomes larger than $d_{boa}/2$, it means that the reference data are not tracking the same optima, and we cannot predict the next optima from the reference data. In terms of prediction severity, we define a threshold T_H. Prediction severity is reliable while below the threshold. Because the predicted possible region is valuable only when the predicted optimal solutions are in the same region, the threshold is set to $d_{boa}/2$, the same as reference severity.

IV. PERFORMANCE EVALUATION

In this study, all six dynamic optimization functions from GDBG are used to evaluate the performance of QBLDPSO. The six test problems are defined in [9]–[11].

F1. Rotation peak function.
F2. Composition of Sphere's function.
F3. Composition of Rastrigin's function.
F4. Composition of Griewank's function.
F5. Composition of Ackley's function.
F6. Hybrid composition function.

Only F1 is a maximization problem, and all the others are minimization problems. In order to provide an intuitive illustration, we transformed F1 into a minimization problem as well. In each function, we tested six types of change from GDBG; the types were selected because they have the highest probability of becoming real problems. Our goal is to use QBLDPSO to solve real problems. Thus, QBLDPSO is designed to cope with these six change types. The framework for the six types of change is described in Fig. 4.

In Fig. 4, $||\varphi||$ denotes the change range of φ, $\varphi_{severity}$ is a constant number that indicates the change severity of φ, φ_{min} is the minimum value of φ, and $noisy_{severity}$ denotes a noise severity between zero and one. $\alpha \in (0, 1)$ and $\beta \in (0, 1)$ are constant values, which are set to 0.04 and 0.1 respectively, in the GDBG system. In our experiments, all parameters are the same as in [9]. A logistic function is used in the chaotic change type, where A is a positive constant between one and four. If φ is a vector, the initial values of the items in φ should be different within $||\varphi||$ in the chaotic change. We use P to denote the period of recurrent change and recurrent change with noise, ϕ to denote the initial phase, and r to denote a random number in the range $(-1, 1)$. The function sign(x) returns 1 when x is greater than 0, -1 when x is less than 0, and 0 otherwise. We use N (0, 1) to denote a normally distributed one-dimensional random number with a mean of zero and standard deviation of one. The six types of change (shown in Fig. 4) are represented as T1 through T6, in the order of small step, large step, random, chaotic, recurrent, and recurrent with noise changes. We basically adopt the same parameters as in [9]–[11]. Additionally, we set the change period for GDBG to 100, 300, and 500, in order to examine different levels of abruptness in change with the same test function. In our experiments, the number of changes is at least 60 (when the change period is set to 500). Thus, the total fitness evaluation is set to 30000. Each experiment repeats 30 times with different random seeds. For QBLDPSO and mQSO [15] parameter settings, the number of swarms is set to 10. Each swarm consists of 10 particles. Following the parameter settings in [15], the default quantum particle number is set to 5, the distance threshold for anti-convergence is 3.0, and the radius of the quantum cloud is 1. For other basic PSO parameter settings, we also adopt the best settings proposed by [15]; $c_1 r_1$ and $c_2 r_2$ are set to 2.0, and ω is set to 0.729. For QBL parameter settings, QBL activates after three changes. We will discuss the parameter settings of W_R in QBLQPA in the following sections.

T1: $\Delta\phi = \alpha \cdot \| \phi \| \cdot \gamma \cdot \phi_{severity}$

T2: $\Delta\phi = \| \phi \| \cdot (\alpha \cdot \text{sign}(\gamma) + (\alpha_{max} - \alpha) \cdot \gamma) \cdot \phi_{severity}$

T3: $\Delta\phi = N(0,1) \cdot \phi_{severity}$

T4: $\phi(t+1) = A \cdot \phi(t) \cdot (1 - \phi(t))/\| \phi \|$

T5: $\phi(t+1) = \phi_{min} + \| \phi \| (\sin\left(\frac{2\pi}{P}t + \varphi\right) + 1)/2$

T6: $\phi(t+1) = \phi_{min} + \| \phi \| \frac{\sin\left(\frac{2\pi}{P}t + \varphi\right) + 1}{2} + N(0,1) \cdot noisy_{severity}$

FIGURE 4. Function of six types of change in GDBG.

Because choosing an appropriate performance evaluation for DOPs is a vital and unsolved issue, many different evaluation methods are proposed and discussed in [6]. In our experiments, we use best of generation (BOG), which is the most commonly used approach. BOG is an optimality-based measure for evaluating how well the algorithm finds the solution with the best fitness values. BOG averages the best solutions in each iteration (generation) over several runs. The measure is

$$\text{BOG} = \sum_{i=1}^{R} \sum_{j=1}^{N} \frac{fitness_{ij}}{R * N} \tag{8}$$

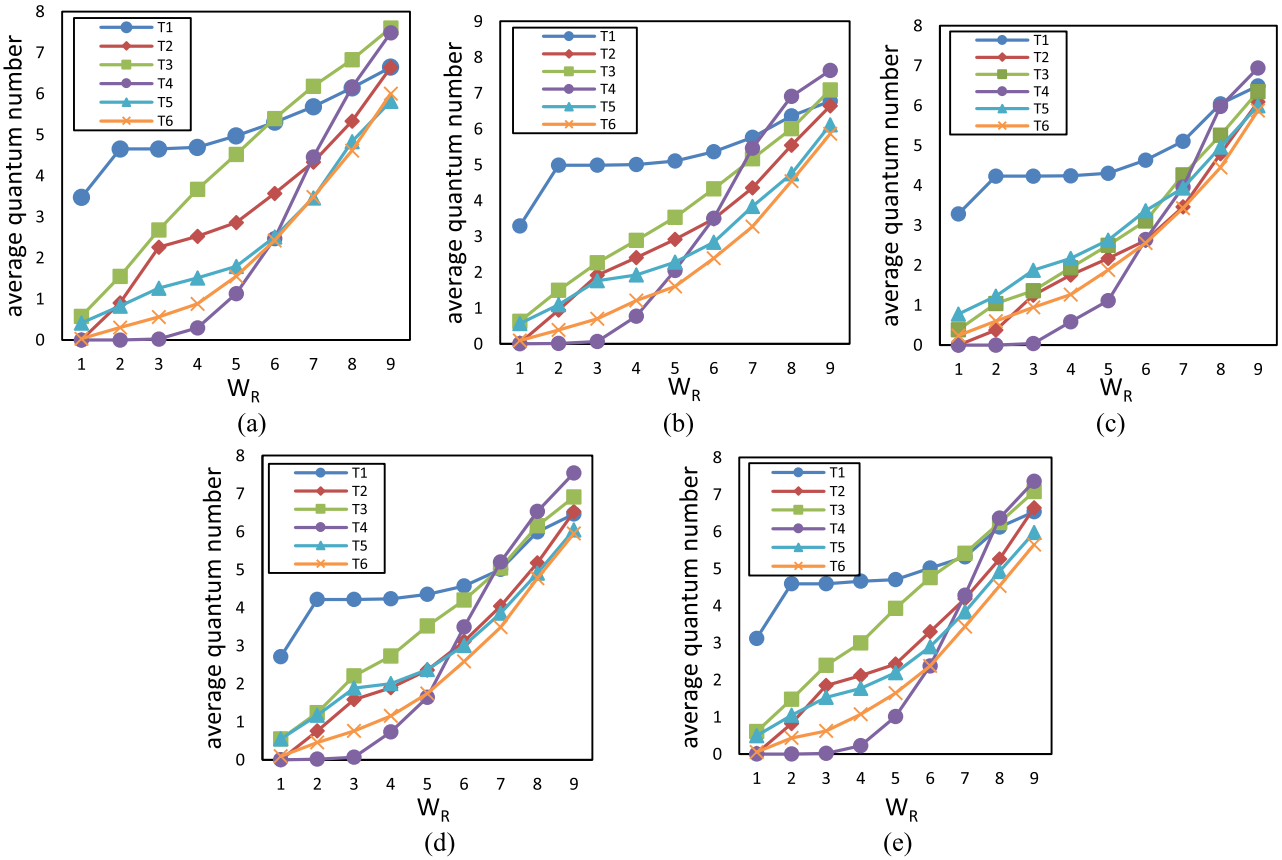


FIGURE 5. GDBG with different change types and different W_R (average quantum number). (a) F1. (b) F2. (c) F3. (d) F4. (e) F5.

where N is the iteration number in each run and R is the run number for each experiment.

QBLDPSO uses two learning strategies: QBLQPA and QBLOP. Each strategy has its own trigger condition. First, we tested the performance of each strategy individually, then we evaluated the overall performance of QBLDPSO. Thus, there are three experiments in this section: the effect of QBLQPA, the effect of QBLOP, and a comparison of QBLDPSO with other algorithms.

A. THE EFFECT OF QBLQPA

In the first experiment, we examined the effect of the parameter W_R in QBLQPA with different kinds of dynamic changes. The parameter W_R defines the effective range of quantum particles. When W_R is large, the acceptable rate of change is relatively high. In this case, there are large number of quantum particles. As a result, the quantum particles around local optimum solutions increase the performance of local exploitation. However, fewer neutral particles lead to reduced overall swarm searching ability. There are fewer quantum particles when W_R is smaller. Thus, the number of neutral particles is relatively large. This means that QBLQPA concentrates on searching all around the swarm rather than tracking the swarm optima.

In order to find the best setting of $W_R$for QBLQPA, we tested W_R values ranging from 1 to 9. Fig. 5 displays the number of quantum particles under varying conditions. The experiments are run using test problems F1 through F5 with change types T1 through T6. The experimental results demonstrate that QBLQPA can adaptively adjust the number of quantum and neutral particles in the swarm based on different environment changes. When the changes are gradual, QBLQPA will understand that the DOP has the characteristic of small displacement. Take T1 (small step changes) for example, the optimal displacement between changes is always within W_R. In this case, the optimal solution should be near the previous optimum. QBLQPA will increase the density of particles around the last optimal solution to search for the new best solution. Thus, the adjusted quantum number in T1 is set larger than for the other change types in Fig. 5. In contrast to T1, the other types of change have a higher chance to exceed W_R. QBLQPA will consider that the optimal solution moves quickly. This displacement is too large for quantum particles to find the optimal solution. Thus, QBLQPA will adapt the number of quantum particles to be smaller.

After testing the effect of the parameter W_R in QBLQPA, we examine the efficiency of QBLQPA using different W_R via BOG. Fig. 6 illustrates the BOG of different change types and W_R compared to mQSO. From the results above (Fig. 6), one can see a range of W_R that is the most appropriate. If W_R is large, the distance between changing optima has

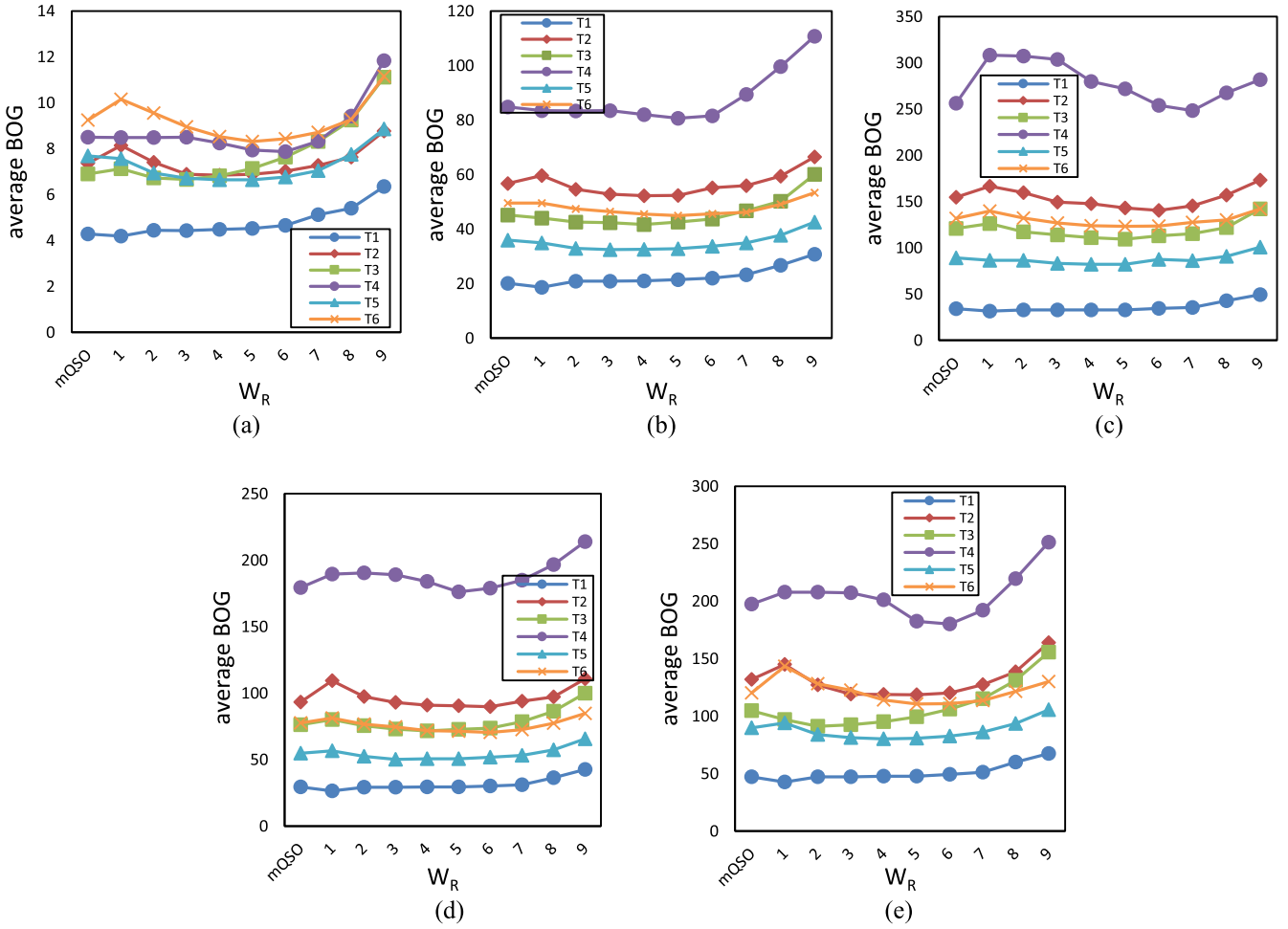


FIGURE 6. GDBG with different change types and different W_R (average BOG). (a) F1. (b) F2. (c) F3. (d) F4. (e) F5.

more probability to remain within W_R, and QBLQPA cannot differentiate the changes. Additionally, an adjustment plan with a large proportion of quantum particles may result in a loss of exploitation ability and efficiency. However, most of the distances between changing optima are larger than W_R if W_R is too small. Once QBLQPA becomes aware of quantum particles becoming inefficient, it changes some of the quantum particles in the swarm to neutral particles to track the moving optima. Fig. 6 indicates that the best setting for W_R is between three and six.

B. THE EFFECT OF QBLOP

We have discussed the effect of parameter W_R in QBLQPA. In this section, we discuss the influence of QBLOP. QBLOP is used to handle environment changes with regular patterns. Thus, we examined QBLOP and compared the results to original mQSO for T5 (recurrent) and T6 (recurrent with noisy changes) problems. The equation for the improvement rate is:

$$\text{Improvement rate1 (IR1)} = \frac{\text{mQSO} - \text{QBLDPSO}}{mQSO}. \quad (9)$$

Fig. 7 displays the experimental results. In T5, QBLOP can predict accurate optima to guide the algorithm and solve the problem more efficiently with a 10% to 40% improvement. This means that QBLOP is able to exploit the regular patterns in the historical data because optimal solutions in the same environment state appear in the same location. For T6, the changing optima have no recurrent pattern, so QBLOP is unable to achieve improvements for this type of environment change.

In our experiment, the noise severity with the parameters from [9]–[11] is 0.8. The characteristics of T5 mean that optimal solutions reoccur in exactly the same place after several changes. In contrast, the optimal solutions of T6 change between several states and reappear in different places. The moving trajectory of T6 is large as a result of the noise. Therefore, QBLOP has difficulty recognizing the recurrent pattern in the optima.

C. COMPARISON WITH OTHER ALGORITHMS

We have focused on improving multi-population based PSO. mQSO and its variant, mPSODE, are two state-of-the-art

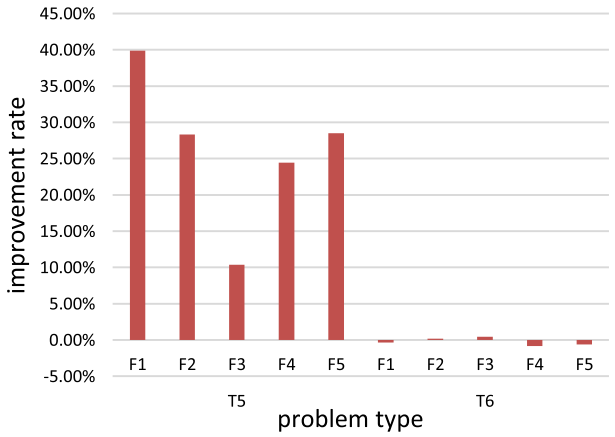


FIGURE 7. Improvement rate of QBLOP and mQSO with different functions and change types.

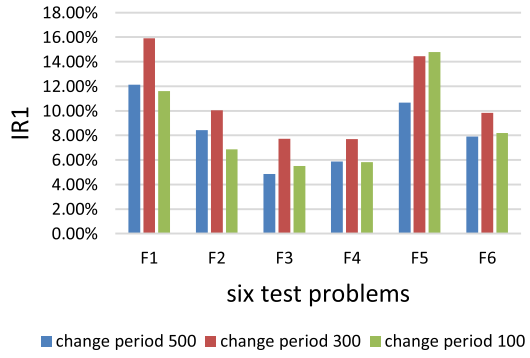


FIGURE 8. Average improvement rate of QBLDPSO over mQSO with the six test problems.

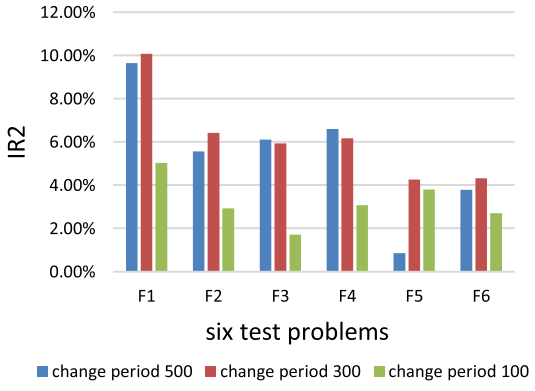


FIGURE 9. Average improvement rate of QBLDPSO over mPSODE with the six test problems.

multi-population based PSO methods. Thus, mQSO and mPSODE are used as the baseline algorithms to compare with QBLDPSO. We determined the parameters for QBLDPSO based on the results of the previous experiments. Thus, W_R is set to five. The improvement rate below is calculated from the results of equation (9):

$$\text{Improvement rate2 (IR2)} = \frac{\text{mPSODE} - \text{QBLDPSO}}{mPSODE}. \quad (10)$$

The average improvement rates of all six test problems from GDBG with different change periods are shown in Fig. 8 and

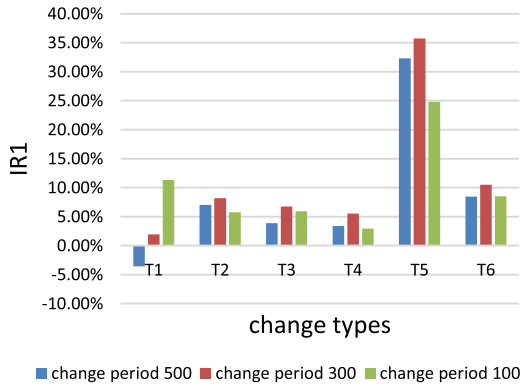


FIGURE 10. Average improvement rate of QBLDPSO over mQSO with different change types.

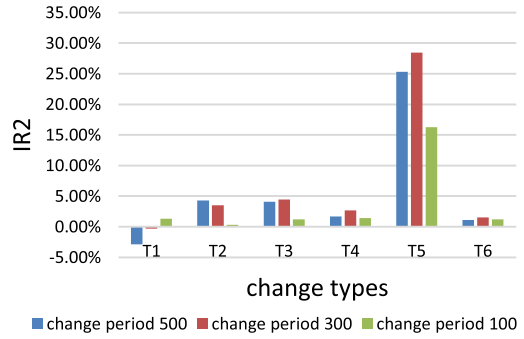


FIGURE 11. Average improvement rate of QBLDPSO over mPSODE with different change types.

Fig. 9. Additionally, the average improvement rates of all different change types are shown in Fig. 10 and Fig. 11. QBLDPSO achieves significant improvements when the change period is 300. These results indicate that neither large nor short change periods are suitable for QBLDPSO. QBLDPSO cannot learn sufficient data in a short change period, and a large change period will cause QBLDPSO to learn too much data, leading to lower performance. In our experiments, we demonstrate that QBLDPSO outperforms mQSO and mPSODE in the GDBG system. QBLDPSO can effectively recognize and respond to changes. In particular, QBLDPSO achieves significant improvements in recurrent change problems.

V. CONCLUSIONS AND FUTURE WORK

We aimed to use QBLDPSO to solve DOPs based on their problem characteristics. Two main issues have been discussed and tested in this study: the ability of QBLDPSO to respond to a changing environment and the improvements achieved by QBLDPSO. We proposed QBLQPA and QBLOP in the oracle algorithms for QBLDPSO. Based on our experimental results, QBLQPA and QBLOP can cope with various dynamic characteristics in DOPs. By correctly tuning its parameters, QBLQPA can cope with abrupt changes. On the other hand, QBLOP can cope with predictable changes by predicting a movement path based on historical data and the environment. In our experiments, QBLDPSO was able

to achieve the best results for various types of DOPs and successfully improve the efficiency of multi-population based PSO in dynamic environments.

To summarize, our study enhances the academic understanding of the characteristics of DOPs through the following features:

1. QBLDPSO combines the QBL mechanism with PSO to solve DOPs.
2. QBLQPA and QBLOP are proposed in the oracle for QBLDPSO based on varying dynamic characteristics.
3. QBLQPA can improve efficiency for both gradual and abrupt changes.
4. For recurrent changes, QBLOP is applied to solve problems by learning historical data.

In the future, QBLDPSO should be applied to more diverse academic environments and realistic applications. The oracle of QBLDPSO could also be given additional characteristics. For example, penalizing a suggestion if the suggestion has previously misguided the optimization process, and adaptively adjusting the quantum radius when movement between environment changes is unstable. Additionally, further studies could combine the QBL mechanism with other optimization algorithms.

REFERENCES

[1] E. Vellasques, R. Sabourin, and E. Granger, "Gaussian mixture modeling for dynamic particle swarm optimization of recurrent problems," in *Proc. 14th Annu. Conf. Genetic Evol. Comput.*, 2012, pp. 73–80.

[2] R.-I. Chang, L.-B. Lai, W.-D. Su, J.-C. Wang, and J.-S. Kouh, "Intrusion detection by backpropagation neural networks with sample-query and attribute-query," *Int. J. Comput. Intell. Res.*, vol. 3, pp. 6–10, Jan. 2007.

[3] R.-I. Chang, S.-Y. Lin, and Y. Hung, "Particle swarm optimization with query-based learning for multi-objective power contract problem," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3116–3126, 2012.

[4] H. Ben-Romdhane, E. Alba, and S. Krichen, "Best practices in measuring algorithm performance for dynamic optimization problems," *Soft Comput.*, vol. 17, no. 6, pp. 1005–1017, 2013.

[5] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 6, pp. 1–24, Oct. 2012.

[6] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: A survey on problems, methods and measures," *Soft Comput.*, vol. 15, no. 7, pp. 1427–1448, 2011.

[7] I. Moser and R. Chiong, "Dynamic function optimization: The moving peaks benchmark," in *Metaheuristics for Dynamic Optimization*. Berlin, Germany: Springer, 2013, pp. 35–59.

[8] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. Congr. Evol. Comput. (CEC)*, Jul. 1999, p. 1882.

[9] C. Li, S. Yang, and D. A. Pelta, "Benchmark generator for the IEEE WCCI-2012 competition on evolutionary computation for dynamic optimization problems," Brunel Univ., London, U.K., Tech. Rep. 2011, 2011.

[10] M. Mavrovouniotis, C. Li, S. Yang, and X. Yao, "Benchmark generator for the IEEE WCCI2014 competition on evolutionary computation for dynamic optimization problems: Dynamic travelling salesman problem benchmark generator," De Montfort University, Leicester, U.K., Tech. Rep. 2013, 2013.

[11] C. Li and S. Yang, "A generalized approach to construct benchmark problems for dynamic optimization," in *Proc. AsiaPacific Conf. Simulated Evol. Learn.*, 2008, pp. 391–400.

[12] T. M. Blackwell, "Swarms in dynamic environments," in *Proc. Genetic Evol. Comput.-Gecco*, 2003, pp. 1–12.

[13] T. Blackwell and J. Branke, "Multiswarm optimization in dynamic environments," in *Proc. Workshops Appl. Evol. Comput.*, 2004, pp. 489–500.

[14] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for dynamic optimization problems," in *Proc. Workshops Appl. Evol. Comput.*, 2004, pp. 513–524.

[15] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, Aug. 2006.

[16] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *Proc. 8th Annu. Conf. Genetic Evol. Comput.*, 2006, pp. 51–58.

[17] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Inf. Sci.*, vol. 178, no. 15, pp. 3096–3109, 2008.

[18] A. B. Hashemi and M. R. Meybodi, "A multi-role cellular PSO for dynamic environments," in *Proc. 14th Int. CSI Comput. Conf. (CSICC)*, 2009, pp. 412–417.

[19] P. Novoa-Hernández, C. C. Corona, and D. A. Pelta, "Efficient multi-swarm PSO algorithms for dynamic environments," *Memetic Comput.*, vol. 3, pp. 163–174, Oct. 2011.

[20] I. G. Del Amo, D. A. Pelta, J. R. González, and A. D. Masegosa, "An algorithm comparison for dynamic optimization problems," *Appl. Soft Comput.*, vol. 12, no. 10, pp. 3176–3192, 2012.

[21] L. Changhe, Y. Shengxiang, and Y. Ming, "Maintaining diversity by clustering in dynamic environments," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2012, pp. 1–8.

[22] K. R. Harrison, B. M. Ombuki-Berman, and A. P. Engelbrecht, "A radius-free quantum particle swarm optimization technique for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2016, pp. 578–585.

[23] A. Simões and E. Costa, "Evolutionary algorithms for dynamic environments: Prediction using linear regression and Markov chains," in *Proc. Parallel Problem Solving From NatureŰPPSN X*, 2008, pp. 306–315.

[24] Z.-J. Wang, Z.-H. Zhan, K.-J. Du, Z.-W. Yu, and J. Zhang, "Orthogonal learning particle swarm optimization with variable relocation for dynamic optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2016, pp. 594–600.

[25] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[26] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proc. Congr. Evol. Comput.*, May 2001, pp. 101–106.

[27] 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.

[28] T. Blackwell, "Particle swarm optimization in dynamic environments," in *Evolutionary Computation in Dynamic and Uncertain Environments*. Berlin, Germany: Springer, 2007, pp. 29–49.

[29] S. Bird and X. Li, "Enhancing the robustness of a speciation-based PSO," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2006, pp. 843–850.

[30] R.-I. Chang and P.-Y. Hsiao, "Unsupervised query-based learning of neural networks using selective-attention and self-regulation," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 205–217, Mar. 1997.

[31] R.-I. Chang, C.-C. Chu, Y.-Y. Wu, and Y.-L. Chen, "Gene clustering by using query-based self-organizing maps," *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6689–6694, 2010.

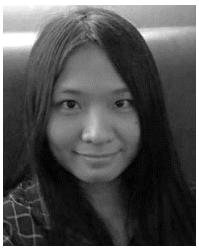
[32] J.-W. Lin, W.-C. Chi, and R.-I. Chang, "Particle swarm optimization combined with querybased learning using MapReduce," in *Future Information Technology*. Berlin, Germnay: Springer, 2014, pp. 91–97.

[33] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Dec. 1995, pp. 1942–1948.

RAY-I CHANG (M'93) is currently a Professor with the Department of Engineering Science and Ocean Engineering, National Taiwan University, Taiwan. He has authored over 200 original papers in international conferences and journals. His current research interests include machine learning, multimedia networking, and data mining. He is a member of CERP and IICM.

HUNG-MIN HSU is currently a Ph.D. Student with National Taiwan University and a Research Assistant with the Research Center for the Information Technology Innovation. He joined the Taiwan e-Learning and Digital Archives Program from 2009 to 2013 and the Academia Sinica Digital Center: System Management and Content Retrieval Technologies for supporting Cloud-based Digital Archive Systems and Services Project from 2014 to 2015. His research interests cover text mining, machine learning, mobile commerce, and related applications.

SHU-YU LIN received the B.S. degree in information management from National Central University in 2005, and the M.S. and Ph.D. degrees in engineering science and ocean engineering from National Taiwan University in 2007 and 2015, respectively. He joined the Institute of Information Science, Academia Sinica, as a Software Engineer in 2009, and is promoted to Project Manager in 2011. His research interests include artificial intelligence, machine learning, data mining, and multimedia.



CHU-CHUN CHANG received the master's degree in engineering science and ocean engineering (computer science) from National Taiwan University. Her research includes data mining and machine learning.

JAN-MING HO received the Ph.D. degree in electrical engineering and computer science from Northwestern University in 1989. He joined the Institute of Information Science, Academia Sinica, as an Associate Research Fellow in 1989, and was promoted to Research Fellow in 1994. He visited the IBM's T. J. Watson Research Center in 1987 and 1988, the Leonardo Fibonacci Institute for the Foundations of Computer Science, Italy, in 1992, and the Dagstuhl Seminar on Applied Combinatorial Methods in VLSI/CAD, Germany, in 1993. From 2004 to 2006, he had served as the Director General of the Division of Planning and Evaluation, National Science Council.