G Model
ASOC 3409 1–22
ARTICLE IN PRESS

Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A new Reinforcement Learning-based Memetic Particle Swarm Optimizer

Q1 Hussein Samma [a,b], Chee Peng Lim [c,*], Junita Mohamad Saleh [a]

[a] School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Malaysia
[b] Faculty of Education, University of Aden, Shabwah, Yemen
[c] Centre for Intelligent Systems Research, Deakin University, Australia

ARTICLE INFO

Article history:
Received 1 August 2015
Received in revised form 2 January 2016
Accepted 5 January 2016
Available online xxx

Keywords:
Memetic algorithm
Particle Swarm Optimization
Reinforcement learning
Local search

ABSTRACT

Developing an effective memetic algorithm that integrates the Particle Swarm Optimization (PSO) algorithm and a local search method is a difficult task. The challenging issues include when the local search method should be called, the frequency of calling the local search method, as well as which particle should undergo the local search operations. Motivated by this challenge, we introduce a new Reinforcement Learning-based Memetic Particle Swarm Optimization (RLMPSO) model. Each particle is subject to five operations under the control of the Reinforcement Learning (RL) algorithm, i.e. exploration, convergence, high-jump, low-jump, and fine-tuning. These operations are executed by the particle according to the action generated by the RL algorithm. The proposed RLMPSO model is evaluated using four uni-modal and multi-modal benchmark problems, six composite benchmark problems, five shifted and rotated benchmark problems, as well as two benchmark application problems. The experimental results show that RLMPSO is useful, and it outperforms a number of state-of-the-art PSO-based algorithms.

1. Introduction

Q3

Memetic-based optimization algorithms have been used successfully in many applications, e.g. DNA sequence compression [1], flow shop scheduling [2], multi-robot path planning [3], wireless sensor networks [4], finance applications [5], image segmentation [6], and radar applications [7]. The main objective of developing memetic-based algorithms is to exploit the benefits of both global and local search methods and combine them into a single model. As an example, the Particle Swarm Optimization (PSO) algorithm is an effective global optimizer, and has been integrated with different local search methods to produce a number of memetic PSO-based models [1,2,8–11]. The resulting models combine the global search strength of PSO and the refinement capability of local search methods into a unified framework.

In the literature, many successful applications of memetic PSO-based models have been reported. In [1], a memetic model integrating PSO and an Intelligent Single Particle Optimizer (ISPO) [12] to solve the DNA sequence compression problem was presented. In [11], an adaptive memetic algorithm with PSO was developed and applied to the Latin hypercube design problem. Specifically, the standard PSO algorithm was adopted to perform the global search operations. It was integrated with a Lamarckian algorithm to perform the refinement operations. A hybrid model of PSO and a pattern-based local search method was studied in [10]. The resulting model was useful for parameter tuning of the Support Vector Machine (SVM). On the other hand, some studies indicate that PSO can be used for performing the local search operations in memetic models [5,13,14]. In [5], a hybrid model of PSO and genetic algorithm was introduced, whereby the PSO algorithm acted as a local search method. A hybrid shuffled frog-leaping algorithm and modified quantum-based PSO local search method was described in [13]. Recently, a hybrid model combining the differential evaluation algorithm and PSO was introduced. Again, PSO functioned as a local search method [14].

There are a lot of challenges in developing an effective memetic-based PSO model. The key challenges include when the local search method should be called, the frequency of calling the local search method, and which particle should undergo the local search operations. Indeed, the findings in [1] indicate that efficient management of the local search method in terms of time and frequency of calling has a significant impact on the performance. Besides these challenges, the standard PSO algorithm also suffers from several weaknesses, primarily the premature convergence and high computational cost problems. The first weakness is related to its fast premature convergence condition [15,16]. As pointed in [15,16], PSO can be trapped quickly in local optima at the beginning of the search process. The second limitation of PSO comes from its high

Q2 * Corresponding author. Tel.: +61 352273307.
E-mail addresses: chee.lim@deakin.edu.au, cplim123@yahoo.com (C.P. Lim).

http://dx.doi.org/10.1016/j.asoc.2016.01.006
1568-4946/

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

computational cost. While a large particle population size gives a better swarm diversity capability, the computational cost becomes intensive too, e.g. each particle needs to undergo the fitness evaluation in every search cycle. This limitation of PSO has been reported in [17,18].

To mitigate the aforementioned problems, this study introduces a new reinforcement learning-based memetic PSO (RLMPSO) model. RL has been employed with standard PSO and other evolutionary algorithms [3,19]. An integration of RL and PSO was Q4 proposed by Grigoris [19]. Another recent study [3] employed RL for parameter tuning a differential evolution algorithm. On the other hand, RL worked independently from PSO in [6], whereby it was adopted to enhance the estimation of the objective function in noisy problems.

Comparing with the existing work in the literature, this study differs in the aspect that RL is embedded in RLMPSO to control the operation of each particle during the search process. Each particle, under the control of RL, performs one of the five possible operations [20], i.e. exploration, convergence, high-jump, low-jump, and fine-tuning. Moreover, each operation is given a reward or penalty according to its achievement. The proposed RMLPSO model has the following advantages:

(1) RLMPSO works with a small population size (typically 3 particles). It utilizes the ISPO (i.e. Intelligent Single Particle Optimizer) algorithm [12]. Additionally, it is enhanced with a total of five operations, i.e. exploration, convergence, high-jump, low-jump, and fine-tuning.
(2) The RL algorithm is embedded into RLMPSO to control the operation of each individual particle in the swarm. Specifically, RL adaptively switches the particle from one operation to another in accordance with the particle's achievement. Positive rewards are given to particles that have performed well, while penalties are imposed to non-performing particles.
(3) Each particle in RLMPSO evolves independently, e.g. one particle executes exploration, while others perform their respective operations.
(4) To minimize the computational cost of fine-tuning, two parameters are introduced i.e. delay (D) and cost (C). The delay parameter prevents fine-tuning (i.e., for local search) to be initiated at the beginning of the search process. The cost parameter controls the duration between each consecutive call of the fine-tuning operation.

Similar to RL, the idea of selecting the best performing operators from a set of alternatives has been comprehensively studied in the literature [21–24]. As an example, four PSO velocity updating strategies were used in [21]. A probability execution variable was assigned for each strategy, and the best operation was given a higher probability of selection. An evolutionary-based optimization algorithm with an ensemble of mutation operators was introduced in [22]. Each individual in the population would select a mutation strategy according to a probability distribution. Improved results were achieved with the ensemble strategy as compared with the single mutation strategy [25].

Differential Evolution (DE)-based methods with ensemble strategies were studied in [23,24,26]. In [23], an evolving DE model with an ensemble mutation strategy was presented. During the search process, DE randomly selected a mutation strategy with a random set of parameters to generate a new offspring. If the produced vector was better than the parent, the strategy would be retained; otherwise a new random mutation strategy with a new set of parameters would be generated [23]. The multi-objective DE algorithm with a pool of Neighbourhood Size (NS) parameter was presented in [24]. In particular, DE was developed using k NS candidates. The best NS value was adaptively selected from k candidates according to their historical performances. Improvements were achieved using k NS candidates as compared with only one candidate. Another DE-based model with an ensemble mutation strategy was presented in [26]. In particular, the population was randomly divided into three small sub-populations and one large sub-population. The three small sub-populations were executed for a specific number of Fitness Evaluations (FEs). Each sub-population was executed with a different mutation strategy, i.e. "current-to-pbest/1" and "current-to-rand/1", and "rand/1" [26]. A reward was computed as the ratio of fitness improvement to the total number of fitness calls consumed by each sub-population. After that, the large sub-population was executed with the setting of the best performing small sub-population. This process was repeated until the maximum number of FEs is met. In this case, the best mutation strategy could be selected dynamically during run time. The proposed model was able to outperform other DE variants.

The rest of this paper is organized as follows. In Section 2, an overview of PSO and its variants is given. The proposed RLMPSO model is explained in Section 3. In Section 4, a series of experiments to evaluate the effectiveness of RLMPSO using benchmark optimization problems is described. A summary of the research findings is presented in Section 5.

2. Particle Swarm Optimization and its variants

PSO was introduced by Kennedy and Eberhart about two decades ago [27]. The motivation of PSO is to mimic social interaction and search behaviours of animals, such as bird flocking and fish schooling. In general, PSO is represented by a swarm of N particles. Each particle in the swarm is associated with two vectors, i.e., the velocity (V) and position (X) vectors, as follows:

$$X_i = \left[d_i^1, d_i^2, d_i^3, \ldots, x_i^D\right] \tag{1}$$

$$V_i = \left[v_i^1, v_i^2, v_i^3, \ldots, v_i^D\right] \tag{2}$$

where D represents the dimension of the optimization problem and i denotes the particle number in the swarm. During the search process, the velocity and position vectors are updated as follows:

$$\begin{aligned} V_{i+1} &= w * V_i + c_1 * rand_{uniform}(pBest - X_i) \\ &\quad + c_2 * rand_{uniform}(gBest - X_i) \end{aligned} \tag{3}$$

$$X_{i+1} = X_i + V_{i+1} \tag{4}$$

where w is the inertia weight, c_1 is the cognitive acceleration coefficient, c_2 is the social acceleration coefficient, $rand_{uniform}$ is a uniformly distributed random number within [0, 1], $pBest$ is the local best position achieved by a particular particle, and $gBest$ is the global best position achieved by the whole swarm.

As can be seen in Eq. (3), each particle's movement is affected by three components, namely its particle velocity (V_i), the distance from its local best ($pBest - X_i$), and the distance from the global best ($gBest - X_i$) in the swarm. Therefore, to control each component in Eq. (3), three parameters are used, i.e., w, c_1, and c_2. The suggested range of the inertia weight is $w \in [0.4, \quad 0.9]$ [27]. It has been pointed out that w must be high in the exploration stage and low in the convergence stage [20]. On the other hand, the settings of c_1 and c_2 need to strike a balance between $pBest$ and $gBest$. As suggested in [20,21], c_1 must be higher than c_2 in the exploration stage, and the opposite in the convergence stage.

Since the introduction of the original PSO algorithm, many PSO variants have been put forward to improve its performance [1,2,4–11,13,14,17,18,28–48]. The main PSO-based algorithms available in the literature can be divided into five categories i.e.

modified-based, hybrid-based, cooperative-based, micro-based, and memetic-based algorithms. A discussion of each category is presented, as follows.

2.1. Modified PSO-based algorithms

The main aim of this category is to enhance the performance of PSO by controlling its parameters [20,45,48], balancing between the exploitation and exploration operations [46,49], or modifying the swarm topology connectivity [44,47,50–53]. An adaptive PSO algorithm was introduced in [20]. The aim was to improve the search performance by efficiently controlling its parameters, i.e. the cognitive acceleration parameter, social acceleration parameter, and inertia weight parameter. These parameters were adaptively tuned by using fuzzy rules [20]. However, the method relied on the distribution of the swarm particles at run time, which was not suitable for PSO with small populations. The PSO variant proposed in [45] assigned independent parameters for each particle in the swarm. Specifically, each particle was given its own parameters (i.e. cognitive acceleration, social acceleration, and inertia weight), and they were tuned adaptively according to the behaviour of the particle during the search process [45]. However, managing these parameters independently would increase the complexity of PSO. Inspired by control theory, a new strategy for controlling the PSO parameters was suggested in [48]. The strategy adopted the concept of the peak time and overshoot in its search process. Nevertheless, the strategy worked with a large population size (i.e. 250 particles); therefore increasing the computational cost owing to fitness evaluation for each particle during each search cycle.

Other researchers attempted to improve PSO by balancing exploitation and exploration operations at run time [46,49]. According to [54], exploration was concerned with spreading the swarm particles to visit the whole search space of the optimization problem, while exploitation was concerned with searching around those visited regions found during the exploration process [54]. The idea of evolving two concurrent swarms as a master–slave model was presented in [46]. The master particles were responsible for exploration while the slave particles performed exploitation. Again, evolving two swarms simultaneously increased the complexity of the proposed model [46]. An intelligent scheme which divided the whole swarm into two groups i.e. exploration and exploitation, was proposed in [49]. Two metrics were developed to split the population, i.e. population spatial diversity and population fitness spatial diversity. Besides its model complexity, the reported results [48] were not competitive on difficult, complex benchmark problems as compared with those from other PSO-based models.

Methods to modify the swarm topology were examined in [44,47,50–53]. The swarm topology is related to the information link between each particle and its neighbour. These links produce different types of topologies, such as fully connected topology [52], ring topology [47], and wheels topology [50]. The dynamics of these topologies can be either static or dynamic. In the former, the topology is fixed during the search process. The latter has dynamically changing topologies at run-time. The main advantage of a dynamic topology over a static one is its ability to prevent the swarm from the premature convergence problem [51] at the beginning of the search process. However, it increases the computational cost required to manage the swarm topology during the search process. In [51], a dynamic topology was proposed. The connection between particles started with one particle, i.e. each particle was connected to another randomly selected particle in the swarm. Then, the connection was linearly increased with time until it reached a fully connected topology, i.e. all particles in the swarm were connected together so that learning from the global best particle in the swarm could take place [51].

One of the drawbacks of modified PSO-based algorithms is the lack of optimization refinement capabilities. In other words, these methods cannot fine-tune the individual dimension of the particles independently without affecting other dimensions.

2.2. Hybrid PSO-based algorithms

Enhancing the effectiveness of PSO by hybridization with other meta-heuristic optimization algorithms has been studied in the literature [34–39,55–58]. PSO has been used to form hybrid models with Ant Colony Optimization (ACO) [37,55,58], GA [34,35,56], Artificial Bee Colony (ABC) [36], and Harmony Search (HS) [38,39]. The main aim of hybrid PSO-based algorithms is to combine the strengths of the constituents into one integrated model. An integration of PSO and HS was proposed in [39]. The PSO swarm was divided into several dynamic multi-swarm particle optimizers, and each sub-swarm was managed by HS. The sub-swarms communicated with one another and exchanged their knowledge after a pre-defined number of FEs. Nevertheless, the model [39] worked with a large population size (i.e. 10 sub-swarms), and required the FEs operation for each search iteration. Another hybrid model of PSO and HS was proposed in [38]. The pitch adjustment operation in HS was replaced with the particle velocity addition operation. The resulting model was applied to dynamic load dispatch optimization problems. As PSO and HS both performed global search [38], the search refinement capabilities were inadequate.

Hybrid PSO and ACO models were studied in [37,55,58]. The hybrid model in [58] comprised two sequential phases, i.e. the ant colony phase and the PSO phase. In addition, a global best exchange operation was added after each search cycle. The proposed model [58] was computational expensive owing to the execution of two swarms simultaneously. Another hybrid PSO and ACO model was developed in [55]. The model [55] comprised four hybridization strategies, i.e., sequential, parallel, sequential with an enlarged table, and a global best exchange strategy. It was proposed to tackle data clustering problems, and the results [55] showed that the hybrid PSO-ACO model outperformed its constituents (i.e. PSO and ACO).

The main challenge of hybrid-based PSO algorithms is threefold: (i) simultaneous managing multiple swarms and exchanging information between them; (ii) the computational cost of the FEs operation for each swarm; (iii) hybrid-based PSO models mainly comprise global search methods [59], and they lack the capability of performing search refinement.

2.3. Cooperative PSO-based algorithms

Unlike the aforementioned categories where all dimensions of the particle are evolved together, cooperative PSO-based algorithms split the optimization process into several sub-problems. This strategy was examined in [28–33,43], in which the task was split into K sub-problems for simultaneous optimization before combining the results.

A cooperative PSO-based algorithm with application to large-scale optimization problems was proposed in [32]. A new position update scheme was introduced to identify the sub-component size in an optimal manner. Another cooperative PSO-based algorithm was developed in [28] to tackle FPGA (Field Programmable Gate Array) placement problems. The placement task was divided into two sub-problems: logic blocks and I/O (Input/Output) blocks. A cooperative-based algorithm was proposed in [30], in which a new statistical strategy was used to decompose the optimization problem. The aim was to estimate the degree of inter-dependencies pertaining to the optimization variables, and then to include the dependent variables in the same sub-problem [30].

In [43], where the original micro PSO algorithm [17] was used with a new master–slave model. At the beginning of the search process, the problem was decomposed into several sub-problems with small dimensions. Then, each sub-problem was solved by an independent micro PSO population [17], and all sub-populations were executed in parallel. As such, the proposed model [43] could be classified as a cooperative PSO-based model. The results in [43] showed its effectiveness in tackling high dimension problems as compared with those from other PSO variants.

One of the key challenges of cooperative-based PSO algorithms is identifying the best sub-problem size and finding the independent variables to be placed in different sub-problems.

2.4. Micro PSO-based algorithms

To minimize the computational cost of PSO with large population sizes, i.e. the cost associated with the FEs operation of each particle during the search process, micro PSO-based algorithms have been introduced [17,18,41–43]. A micro PSO algorithm was introduced in [17] for tackling high dimension optimization problems. The results [17] showed the capability of the developed optimizer to achieve competitive performance as compared with those from the standard PSO algorithm. On the other hand, the application of micro PSO to image reconstruction problems was investigated in [18]. The limitations of the micro PSO algorithm in [17,18] included prevention from exploitation of possible promising search regions due to the repelling force, as well as the high computational cost of re-starting the whole micro swarm during the search process.

The application of a micro PSO algorithm to real-world design problems was discussed in [41,42]. PSO with a small population size was developed for tuning the parameters of power system stabilizers [42]. A re-generation scheme was used to improve diversity of the micro PSO swarm, where the position and velocity vectors were randomized after a pre-defined number of FEs. The reported results [42] revealed the usefulness of the re-generation strategy in enhancing the diversity property.

In summary, the main advantage of micro PSO-based algorithms is the ability to overcome the high computational cost per each particle in a standard PSO algorithm. However, it lacks population diversity due to the small number of particles employed. It also suffers from the problem of premature convergence, i.e., the population converges rapidly towards the global best particle at the early stage of the search process [17].

2.5. Memetic PSO-based algorithms

As the PSO algorithm is generally a global search optimizer [1], it lacks the capability of refining its local search space. Therefore, a number of investigations to integrate PSO with local search methods to produce memetic-based PSO models have been studied [1,2,4,5,7–11,13,14,60]. A memetic model of PSO and two local search methods, i.e. cognition-based search and random walk, was introduced in [9]. The local search methods were able to enhance the performance of the standard PSO algorithm. Another memetic-based PSO model was studied in [10]. In particular, a probabilistic selection scheme to determine which particle should undergo local search was developed [10]. Then, those particles with better fitness were given a higher probability to be fine-tuned [10]. A hierarchical-based memetic model was developed in [8]. The model contained two (top and bottom) layers. The bottom layer comprised M swarms while the top layer comprised one swarm with M best particles from the bottom layer. In addition, the search process consecutively switched from the bottom layer to the top layer. A Latin hypercube sampling optimizer was embedded for the fine-tuning operation, which was triggered every ten search generations [8]. Nevertheless, managing local search methods in terms of when to call the local search method, as well as the frequency of calling posed as a difficult problem in memetic-based PSO models.

Another memetic-based PSO algorithm was reported in [13]. An improved Quantum-based PSO optimizer was developed to act as a global optimizer, while the shuffled complex evolution technique was adopted for local search operations. The numerical results [13] indicated that the local search method was able to improve the PSO performance as compared with that from the standard PSO algorithm. A recent study of a memetic-based PSO algorithm with application to a large-scale Latin hypercube design problem was presented in [11]. Specifically, PSO was integrated with multiple local search methods to tackle the hypercube design problem. However, incorporating multiple local search methods increased FEs in each local refinement operation [11].

The synergy of PSO with local search methods was discussed in [60]. In particular, an enhanced PSO algorithm was integrated with gradient-based and derivative-free local search methods. The gradient-based methods were used for numerical optimization problems, while the derivative-free local search methods were adopted for real-world problems. The reported results revealed that the employed local search methods were able to improve the search performance of PSO [61].

Real-world applications of memetic-based PSO algorithms were reported in [1,2,4,5,7]. A memetic-based PSO model for undertaking flow shop scheduling problems was described in [2]. Several local search methods were developed, e.g. nawaz-enscore and simulated-annealing, and they were embedded in the memetic-based PSO model. A memetic PSO optimizer for tackling DNA sequence compression problems was presented in [1]. The search space was clustered into several regions for facilitating the local search operations. The frequency of calling the local search method was suggested to be low [1]. For financial applications, an integrated model of GA and PSO was developed in [5]. GA and PSO were used as a global optimizer and a local search method, respectively. The application of a memetic-based PSO algorithm to wireless sensor networks was described in [4]. Operating as a global search optimizer, PSO was integrated with an active-set local search method [4]. The proposed model [4] was used to maximize the quality of transmitted video streams by visual sensors. A recent study of a memetic-based PSO optimizer for radar applications was presented in [7]. The combination of PSO as a global search optimizer and a gradient-free local search method was proposed [7]. A memetic-based PSO optimizer for SVM parameter tuning was examined in [10]. In particular, the standard PSO optimizer was integrated with a pattern-based local search method for refinement operations. The results showed the effectiveness of the memetic-based model in tuning SVM parameters, as compared with those from the standard PSO as well as other optimizers reported in [10].

3. The proposed model

The proposed RLMSPO model integrates RL into the memetic PSO operations. The detailed explanations are as follows.

3.1. Reinforcement learning

RL [62] stems from research in machine learning and artificial intelligence. It has been widely studied in game theory [63,64]. The main components of RL include a learning agent, an environment, states, actions, and rewards. To implement RL in this study, the Q-learning algorithm [65] is adopted.

Let $S = [S_1, S_2, \ldots, S_n]$ be a set of states of the learning agent, $A = [a_1, a_2, \ldots, a_n]$ be a set of actions that the learning agent can execute, r_{t+1} be the immediate reward acquired from executing action a, γ be the discount factor within [0,1], α be the learning rate within [0,1], $Q(s_t, a_t)$ be the total cumulative reward that the learning agent has gained at time t, and it is computed as follows:

$$Q_{t+1}(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma\, max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (5)$$

where, Max_{FEs} is the maximum number of FEs. The discount factor γ is responsible for penalizing the future reward. When $\gamma = 0$, Q-learning considers the current reward only. When $\gamma = 1$, Q-learning looks for a higher, long-term reward. It is suggested to set $\gamma = 0.8$ [3].

Algorithm 1. Q-learning algorithm

```
DO for each state s_t ∈ A = {s_1, s_2, .., s_n} and action a ∈ A = {a_1, a_2, .., a_n}
set Q(s_t, a_t) = zero in the Q-table
END
Randomly select an initial state s_t
Repeat
    Select the best action, a_t, for the current state s_t from the Q-table
    Execute action a_t and get the immediate reward, r
    Get the maximum Q value for the next state s_{t+1}
    Update the Q-table entry using Eq. (5)
    Update the current state, s_t = s_{t+1}
Until the maximum number of FEs is met
```

A numerical example is presented to clarify Eq. (5), as follows. Assuming a learning agent with s_t has to perform one of the four possible actions, i.e. move up, move down, move left, or move right, as shown in Fig. 1. After executing the "move right" action with a reward of 1 (i.e., $r = 1$), the next state is s_{t+1}, as shown in Fig. 1(b).

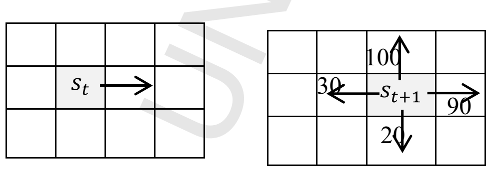
Assume that the parameter settings are as follows: the pervious value stored in the Q-table for $Q(s_t, a_t)$ is 10, i.e. $Q(s_t, a_t) = 10$; the discount factor is 0.1, i.e. $\gamma = 0.1$; and the learning rate parameter is 0.9, i.e., $\alpha = 0.9$. Then, the new value in the Q-table updated to

$$Q_{t+1}(s_t, a_t) = 10 + 0.9 * [1 + 0.1 * \max(20, 30, 100, 90) - 10] = 10.9$$

Then, update the state $s_t \to s_{t+1}$

The search steps of the Q-learning algorithm are illustrated in Algorithm 1. One of the main characteristics of Q-learning is how the learning rate (i.e., α) determines the extent of which the newly learned information overrides the existing, old information. As an example, when α is close to 1, this means that a higher priority is given to the newly gained information, and Q-learning performs more exploration for all defined states. On the other hand, a small α value gives a higher priority for the existing information in the Q-table to be exploited. This puts Q-learning in the exploitation mode. For this reason, α normally is set to a high value at the beginning of the search process, and is decreased at each time step, in order to switch to the exploitation mode, as follows [3]:

$$\alpha(t) = 1 - \left(0.9 * \frac{t}{\text{Max}_{\text{FEs}}}\right) \quad (6)$$



(a) (b)

Fig. 1. A numerical illustration of (a) the current state and (b) the next state.

3.2. The RLMPSO structure

Fig. 2 shows the overall RLMPSO structure that integrates RL and PSO. The PSO particles acts as the RL agents. The environment is characterized by the search space of the particles. The states represent the current operation of each particle, i.e., exploration, convergence, high-jump, low-jump, or fine-tuning. The action is defined as it changes from one state to another. As can be seen in Fig. 2, RL controls the operation of each particle in the PSO swarm. Specifically, RL adaptively switches the particle from one operation (state) to another according to the particle's achievement. Positive rewards are given to particles that have performed well, while penalties (negative rewards) are given to non-performing particles.

In a standard PSO algorithm, the exploration operation is initiated at the beginning for the whole swarm particles. Then, the operation gradually switches to the convergence state towards the end of the search process. In RLMPSO, the choice of the most suitable search operation for each particle is selected adaptively using RL. Algorithm 2 illustrates the proposed RLMPSO search procedure. The procedure is repeated until the maximum number of FEs is met.

The main interaction between Q-learning and the five possible search operations can be summarized in three steps, as follows:

(i) Obtain the best operation to be executed based on the Q-table value for the current particle.
(ii) Execute the selected operation and compute the fitness function. The immediate reward is computed, i.e.,

$$r = \begin{cases} 1 & \text{if fitness is improved} \\ -1 & \text{otherwise} \end{cases} \quad (7)$$

(iii) Update the Q-table for the current particle using Eq. (5).

In Algorithm 2, after calling the fine-tuning operation, a cost (i.e. a negative reward) is given to penalize the execution of this operation, in order to give a higher priority for other operations to be executed (as further clarified in Section 3.7).

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

Algorithm 2: The RLMPSO model $(N, a, p, J, R_{max}, R_{min}, C, D, V_{max}, V_{min}, Max_{FEs})$

Initialize
Randomly generate an initial population of N particles with random position X_i and random velocity V_i.
DO for each particle, X_i
Compute fitness $f(X_i)$, and assign
$pBest_i = X_i$, $f(pBest_i) = f(X_i)$
for each state $s_t \in A = \{s_1, s_2, .., s_5\}$ and action $a \in A = \{a_1, a_2, .., a_5\}$
set $Q(s_t, a_t) = zero$ in the Q-table
END

Assign $gBest = X_i$, where X_i is the fittest particle in the swarm, and *fitcount=0.*
Set the current state s_t for all particles to the exploration state

While *fitcount* <= Max_{FEs}
DO for each particle, X_i
Select the best action, a_t, for the current state s_t from the Q-table
SWITCH action
CASE exploration
Update V_i, *and* X_i using Eqs. (3) and (4)
CASE convergence
Update V_i, *and* X_i using Eqs.(3) and (4)
CASE High or low jump
Update X_i using Eq. (9)
CASE fine-tuning
DO for each dimension of particle $pBest_i$
Repeat *J* times
Update $pBest_i$ using Eq. (10)
Compute fitness $f(pBest_i)$
fitcount = *fitcount* +1
IF *fitcount* > Max $_{FEs}$
EXIT
END
END
END
Set the immediate reward to a negative cost value (r = cost)
END
IF action is not fine-tuning
Compute fitness $f(X_i)$
fitcount = *fitcount* + 1
Get the immediate reward r using Eq. (7)
END

Get the maximum Q value for the next state s_{t+1}
Update the Q-table entry using Eq. (5)
Update the current state, $s_t = s_{t+1}$

IF $f(X_i) < f(pBest_i)$
$pBest_i = X_i$
END
IF $f(X_i) < f(gBest)$
$gBest = X_i$
END
END
END

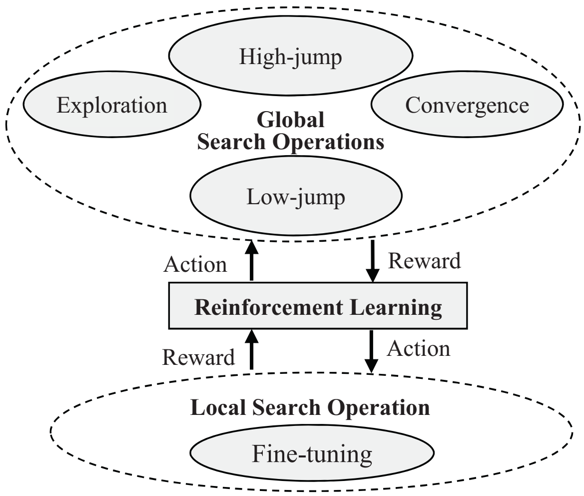


Fig. 2. The proposed RLMPSO structure.

3.3. The Q-table and its contents

The Q-table is shown in Fig. 3. It is an $M \times M$ matrix, where M is the number of states. In RLMPSO, each particle has its own Q-table. Therefore, to minimize the computational cost of managing the Q-tables, a micro PSO model with a small population size (i.e., 3-particles) has been used throughout this research.

To delay the execution of the fine-tuning operation (F) at the beginning of the search process and to give a higher priority for other operations to be executed, the initial Q-table entry for state F is set to a negative value, as indicated in Fig. 3. In addition, RLMPSO has to be executed N times (i.e., a minimum lapse of N FEs is required) before RL activates fine-tuning. Finally, the maximum negative value is considered as the initial value of F, as shown in Fig. 4.

During the execution of RLMPSO, the best action for the current state is retrieved from the Q-table, as follows:

$$best\ action = Max[Q(\ current\ state, \quad all\ actions)] \tag{8}$$

states / action	E	C	H	L	F
E	0	0	0	0	-10.78
C	0	0	0	0	-10.78
H	0	0	0	0	-10.78
L	0	0	0	0	-10.78
F	0	0	0	0	-10.78

Fig. 4. The initial values in the Q-table of particle 1.

A numerical example of the Q-table entries for Particle 1 is shown in Fig. 5. Assuming that the current state of Particle 1 is exploration (E). When Eq. (8) is applied, the next state is C, as indicated in Fig. 6.

To update the content of the Q-table, Eqs. (5) and (6) are used. The new content of the Q-table is shown in Fig. 6. As can be seen, after executing the exploration (E) operation, Particle 1 receives a penalty because it cannot improve the search process.

3.4. The boundary condition

In PSO, there are four possible boundary conditions, i.e. reflecting wall, damping wall, invisible wall, and absorbing wall, as shown in Fig. 7. The details are as follows:

(i) *Reflecting wall*: When a particle exceeds the limit of the search space in any dimension X_i, the sign of its velocity (i.e., V_i) is changed, and X_i is reflected back to the search space.
(ii) *Damping wall*: This case is similar to the reflecting wall except that the particle is reflected with a small random value.
(iii) *Invisible wall*: The particle is allowed to jump out of the predefined search space; however, the fitness function is not computed.
(iv) *Absorbing wall*: When the particle exceeds the limit of the search space in any dimension X_i, its velocity V_i is set to zero, and X_i is set to the boundary limit.

states / action	E	C	H	L	F
E	0	0	0	0	-inf
C	0	0	0	0	-inf
H	0	0	0	0	-inf
L	0	0	0	0	-inf
F	0	0	0	0	-inf

(a) FEs (N=1)

states / action	E	C	H	L	F
E	-3.29	-5.29	-4.52	-4.52	- inf
C	-5.29	-5.99	-5.33	-5.29	- inf
H	-3.40	-4.67	-5.05	-5.29	- inf
L	-4.52	-3.34	-4.67	-5.29	- inf
F	0	0	0	0	- inf

(b) FEs (N=100)

states / action	E	C	H	L	F
E	-9.90	-9.23	-10.78	-8.2	- inf
C	-9.87	-9.88	-10.44	-7.92	- inf
H	-9.84	-9.18	-9.98	-8.34	- inf
L	-9.01	-9.76	-10.14	-10.2	- inf
F	0	0	9 0	0	- inf

(c) FEs (N=1000)

Fig. 3. (a)–(c) computing the initial value of state F.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

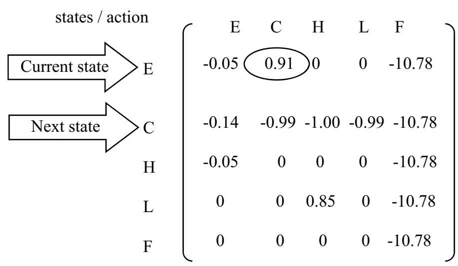


Fig. 5. The Q-table of particle 1 after five operations.

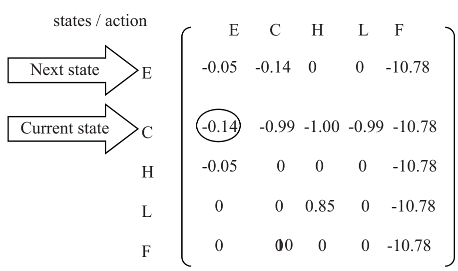


Fig. 6. The Q-table values of particle 1 after six operations.

In this research, the damping wall, which has been used in a number of PSO variants [18,66], is adopted.

3.5. Exploration and convergence operations

The exploration and convergence operations are normally executed at the beginning and towards the end of the search process, respectively. However, some studies recommend switching adaptively at any time from exploration to convergence, and vice versa. Motivated by the findings in [20], RLMPSO can execute any state, i.e. exploration, convergence, high-jump, low-jump, and fine-tuning, at any time during the search process. RL is responsible to keep track of the best executed operation pertaining to each particle.

As stated earlier, particle X_i moves in the search space guided by the global best particle, *gBest*, its current velocity, V_i and the local best particle, $pBest_i$, as indicated in Eq. (3). Parameters w, c_1 and c_2 control the direction and movement of $particle_i$, as shown in Figs. 8 and 9. Therefore, in the exploration state, w should be high to allow the particle to make a large movement to explore the search

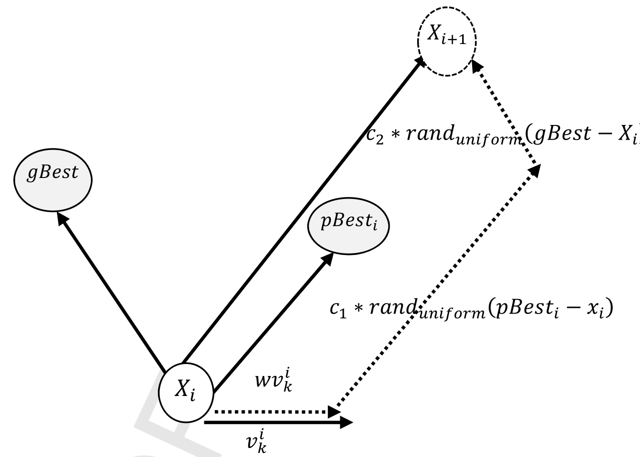


Fig. 8. The exploration operation.

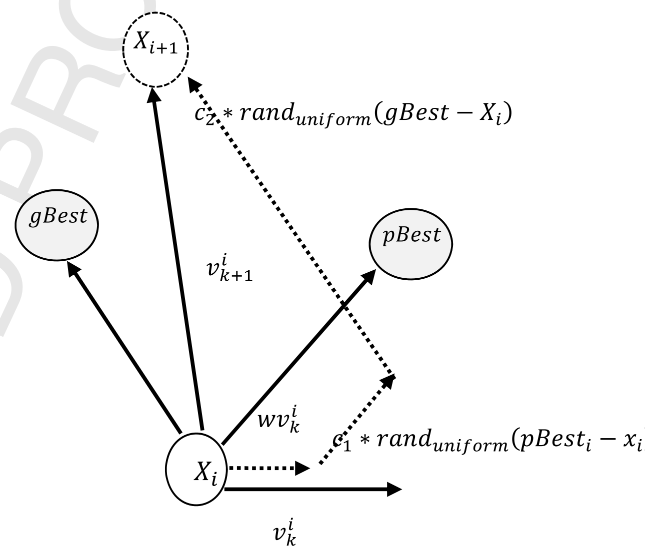


Fig. 9. The convergence operation.

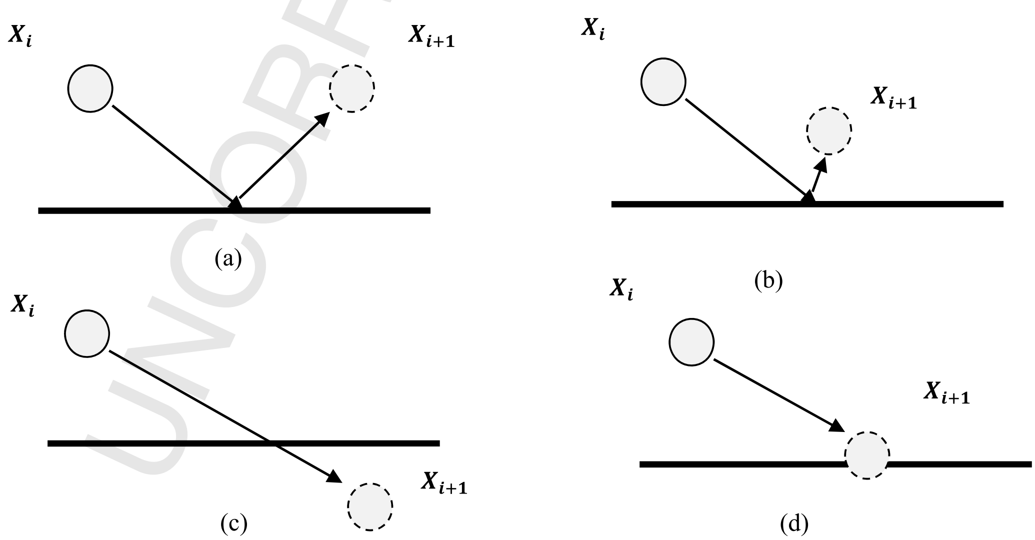


Fig. 7. The boundary conditions of PSO, (a) reflecting wall, (b) damping wall, and (c) invisible wall (d) absorbing wall.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

space. Moreover, as stated in [20], c_1 should be higher than c_2 in the exploration mode, in order to move the particle far away from the global best particle, as shown in Fig. 8.

The convergence operation is similar to the exploration operation, except that all particles converge slowly towards the global best particle, *gBest*. Therefore, *w* should be low, in order to prevent particle X_i from oscillating around the *gBest* location. Moreover, the settings of c_1 and c_2 should be the opposite of those in the exploration mode. In this study, $c_1 = 0.5$ and $c_2 = 2.5$, as used in [67]. Fig. 9 shows the location of particle X_{i+1} after applying Eq. (3).

3.6. High and low jump operations

The high and low jump operations have been used in many PSO-based variants [20,49–51]. The main idea of these jump operations is to enable the local best particle, $pBest_i$, to escape from possible local optima. Specifically, a random value is added to each dimension of $pBest_i$, as follows:

$$X_i = pBest_i + rand_{normal}(R_{\max} - R_{\min}) \tag{9}$$

where R_{max}, and R_{min} are the maximum and minimum boundaries of the search space, respectively, $rand_{normal} \in [0, 1]$ is a normal distributed random number, i.e., $N \sim (u, \sigma^2)$ with mean $u = 0$ and standard deviation σ. Note that $\sigma = 0.9$ helps the escape with a high jump while $\sigma = 0.1$ helps the escape with a low jump.

3.7. Fine-tuning operation

The fine-tuning operation aims to fine-tune each dimension, d_i, of particle $pbest_i$ independently from other dimensions, as indicated in Fig. 10.

In this study, the ISPO model [12] is adopted for the fine-tuning operation. The details of fine-tuning are shown in Algorithm 3. The fine-tuning operation iterates through all dimensions of $pBest_{i,d}$, and it tunes each dimension independently. As indicated in Algorithm 3, the search process continues for *J* times. In ISPO, the velocity is computed as follows [12]:

$$V_{i,d} = \frac{a}{j^p} r + L_{i,d} \tag{10}$$

where *a* is the acceleration factor, *p* is the descent parameter that controls the decay of the velocity, *r* is a uniformly distributed random number within [−0.5,0.5], and *j* is the current FEs number. Variable $L_{i,d}$ represents the learning rate that controls the jumping size. Its value is doubled if the fitness value improves; otherwise it is decreased. As such, $L_{i,d}$ is updated as follows:

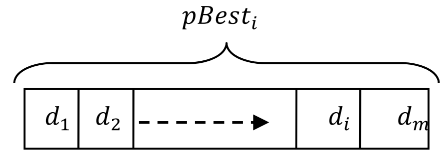


Fig. 10. Fine-tuning operation.

$$L_{i,d} = \begin{cases} 2V_{i,d} & \text{if fitness improved} \\ \dfrac{L_{i,d}}{2} & \text{otherwise} \end{cases} \tag{11}$$

The value of $pBest_{i,d}$ is updated as follows:

$$pBest_{i,d} = \begin{cases} pBest_{i,d} + V_{i,d} & \text{if fitness improved} \\ pBest_{i,d} & \text{otherwise} \end{cases} \tag{12}$$

Algorithm3: The fine-tuning operation ($pBest_i, V_{max}, V_{min}, R_{max}, R_{min}, Max_{FEs}, J, a, p, fitcount$)

Do d=1 to D_{max}
  While (J *FEs* is not met) and ($fitcount < Max_{FEs}$)
    Update $V_{i,d}$ using Eq (12)
    Compute fitness $f(pBest_i)$
    Update $pBest_{i,d}$ using Eq (14)
    Update $L_{i,d}$ using Eq (13)
    fitcount = fitcount+1;
  END
END

Fine-tuning is useful for exploiting promising search regions. However, it has been given a low priority because it consumes a high number of FEs as compared with other operations which take only a single FEs per call, as indicated in Algorithm 2. The execution of fine-tuning must be delayed until the global operations i.e. exploration operation, convergence operation, and jumping operations, have been performed. This allows the fine-tuning operation to perform exploitation of the regions that have been explored by the global search operations. Therefore, to prevent RLMPSO from executing the fine-tuning operation at the beginning of the search process, fine-tuning in the *Q*-table is initialized with a negative value, in order to delay its execution (i.e. after a minimum lapse of *N* FEs, as discussed in section 3.3). Moreover, a cost parameter is introduced to provide an internal delay between consecutive fine-tuning operation calls, as shown in Fig. 11.

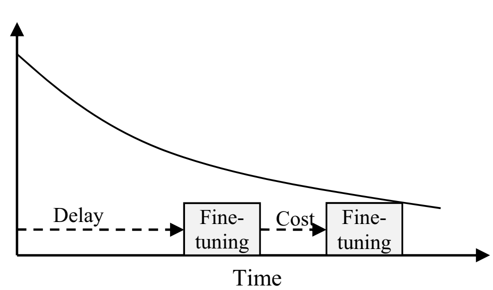


Fig. 11. Delay and cost parameters.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

Table 1
List of parameter settings used in this study.

Parameters	Value
N (Population size)	3 particles
w (Inertia weight)	0.9 for exploration operation 0.4 for convergence operation
c1	2.5 for exploration operation 0.5 for convergence operation
c2	0.5 for exploration operation 2.5 for convergence operation
$[V_{min}, V_{max}]$	0.2 of search range
C, and D	(-2,1000)
ISPO [12]	J = 30 , p=20, a = 150

4. Experimental study

Three benchmark optimization problems were investigated in this experiment, i.e. unimodal, multi-modal, composite problems, shifted, and rotated problems. In addition, two real-world problems were studied. The details are as follows.

4.1. Parameter settings and performance metrics

Table 1 shows the parameter settings of RLMPSO. For performance evaluation, the mean fitness value was computed from the best fitness values obtained from different runs. The convergence curve was computed during the RLMPSO search process.

Table 3
Parameters and levels of the CCD experiment.

Parameter	Level		
	Low (−1)	Medium (0)	High (+1)
D	10 FEs	100 FEs	1000 FEs
C	−2	−4	−8

4.2. Case study I: unimodal and multi-modal benchmark problems

A total of four commonly used unimodal and multi-modal problems, i.e. Sphere, Schwefel, Ackely, and Griewank, were examined. These benchmark problems were studied previously using PSO in [1,68,69]. Table 2 shows the mathematical formula of each benchmark problem as well as the associated search range. The maximum number of FEs was set to $FE_{max} = 2.5 \times 10^5$, as in [1].

4.2.1. Analysis of the delay and cost parameters

The Centre Composite Design (CCD), a useful design-of-experiment method [70,71], was employed to analyze the effects of the delay (D) and cost (C) parameters pertaining to the RLMPSO performance. During the experimental run, the value of each parameter was set at three different levels, i.e. low, medium, and high. The possible combinations of the experimental parameters at each level were generated, in order to study the interaction between these parameters. A total of 100 runs for each of the five cases were carried out with different levels of both D and C settings. Table 3 and Fig. 12 show the detailed parameter settings and experimental configurations.

As can be seen in Table 4, for Exp. 4 (D = 1000 FEs, and C = −2), the cost (penalty) of executing fine-tuning was set at −2 by RL. On the other hand, D = 1000 indicated that the fine-tuning operation was delayed by a negative value of −10.78 (explained in Section 3.3). In other words, a minimum lapse of 1000 FEs was required before fine-tuning could be executed.

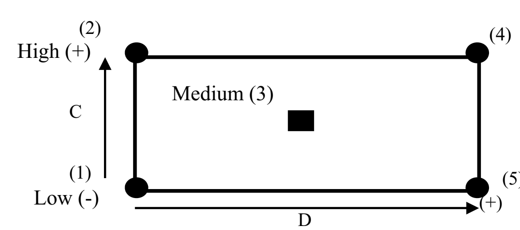


Fig. 12. A CCD experiment with two parameters and five points (i.e. one centre and four corners).

Table 2
Unimodal and multi-modal benchmarks used in this study.

Test function	Mathematical formula	Search range
Sphere	$f_1(x) = \sum_{n=1}^{N} x_n^2$	$-100 \le x_n \le 100$
Schwefel 2.22	$f_2(x) = \sum_{n=1}^{N} \lvert x_n \rvert + \prod_{n=1}^{N} \lvert x_n \rvert$	$-10 \le x_n \le 10$
Ackley	$f_5(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2}\right) - \exp\left(\frac{1}{N}\sum_{n=1}^{N}\cos(2\pi x_n)\right)$	$-32 \le x_n \le 32$
Griewank	$f_3(x) = 1 + \frac{1}{4000}\sum_{n=1}^{N} x_n^2 - \prod_{n=1}^{N}\cos\left(\frac{x_n}{\sqrt{n}}\right)$	$-600 \le x_n \le 600$

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

Table 4
Effects of delay and cost parameters on RLMSPO.

Function	Sphere (Std. dev.)	Schwefel (Std. dev.)	Ackley (Std. dev.)	Griewank (Std. dev.)
Experiment 1 ($D=10$, and $C=-2$)	7.37e−54 (2.11e−53)	2.67e−29 (2.77e−29)	5.14e−14 (8.21e−15)	4.32e−4 (2.14e−05)
Experiment 2 ($D=100$, and $C=-4$)	8.52e−29 (2.69e−28)	1.93e−17 (4.02e−17)	3.89e−08 (1.22e−07)	2.20 e−03 (3.60 e−03)
Experiment 3 ($D=1000$, and $C=-8$)	2.36e−08 (4.98e−08)	4.02e−05 (8.41e−05)	0.3051 (0.4880)	2.0 e−03 (3.60 e−03)
Experiment 4 ($D=1000$, and $C=-2$)	**6.62e−56** **(1.64e−55)**	**2.25e−29** **(9.44e−30)**	**4.81e−14** **(5.51e−15)**	**1.54e−05** **(4.87e−05)**
Experiment 5 ($D=10$, and $C=-8$)	4.12e−15 (1.19e−14)	9.98e−09 (3.06e−08)	0.0025 (0.0075)	2.0 e−03 (3.20 e−03)

For each benchmark problem, the experiment was repeated 100 times. The averages and standard deviations are reported in Table 4. Better results were achieved from two configurations, i.e., $D=1000$ FEs and $C=-2$ as well as $D=10$ FEs and $C=-2$. This implied that RLMPSO could produce better results with small penalty values. The worst result was produced with the highest cost value. On the other hand, the best result was produced by Exp. 4, where the delay value was the highest while the cost value was the lowest. The results in Table 4 reveal that it is better to delay the execution of fine-tuning rather than calling it in the early stage, where it requires a large computational cost for FEs especially in high-dimensional problems.

For further analysis, the average number of calls pertaining to each operation and the average number of FEs for each operation were computed. Fig. 13 shows the results plotted in the logarithm scale. The fine-tuning operation accumulated the lowest number of calls, owing to the restriction of the cost and delay parameters embedded in the RL algorithm. The minimum number of calls pertaining to fine-tuning occurred in Exp. 3, in which the cost and delay parameters were the highest. Other operations showed a similar average number of calls. Fig. 14 shows the average numbers of FEs for each operation. The fine-tuning operation showed the highest value. Exp. 1 and Exp. 4 required the highest FEs in all benchmark problems. This was because Exp. 1 and Exp. 4 had the minimum cost ($C=-2$).

Table 5
The RLMPSO performance with different population sizes.

Function	3 Particles (Std. dev)	5 Particles (Std. dev)	10 Particles (Std. dev)
Sphere	**6.62e−56** **(1.64e−55)**	6.69e−55 (1.60e−54)	6.74e−39 (1.27e−38)
Schwefel	**2.25e−29** **(9.44e−30)**	3.54e−29 (2.96e−29)	4.45e−19 (7.83e−19)
Ackley	**4.81e−14** **(5.51e−15)**	5.17e−14 (1.19e−14)	9.54e−10 (2.51e−09)
Griewank	**1.54e−05** **(4.87e−05)**	5.33e−03 (3.15e−02)	3.7e−03 (3.5e−03)

4.2.2. Analysis of the number of particles

The effect of the population size on the RLMPSO performance was evaluated. Table 5 shows the results of varying the number of particles. A larger population size degraded the results in all benchmark problems. This was owing to the increase in complexity of RLMPSO, i.e., the increase of the number of Q-tables and the FEs pertaining to the fine-tuning operation.

4.2.3. Analysis of the contributions of each operation

The importance of each individual operation in RLMPSO was analyzed. RLMPSO was executed 100 times, each with one of its

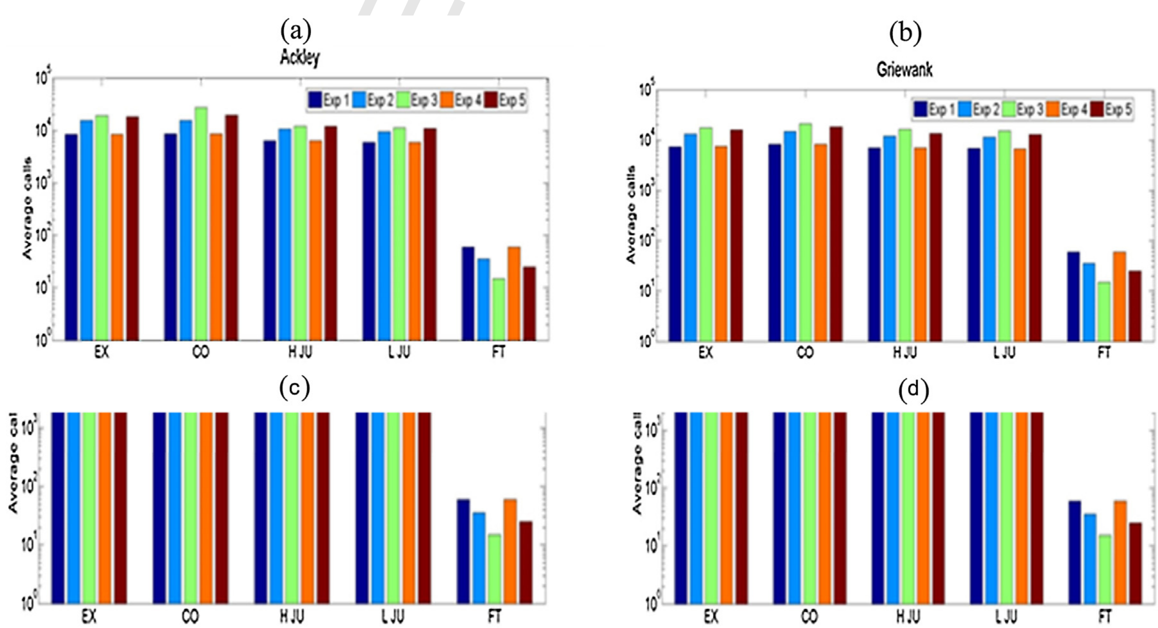


Fig. 13. Average calls of each RLMPSO operation for (a) Sphere, (b) Schwefel, (c), Ackley (d) Griewank functions.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

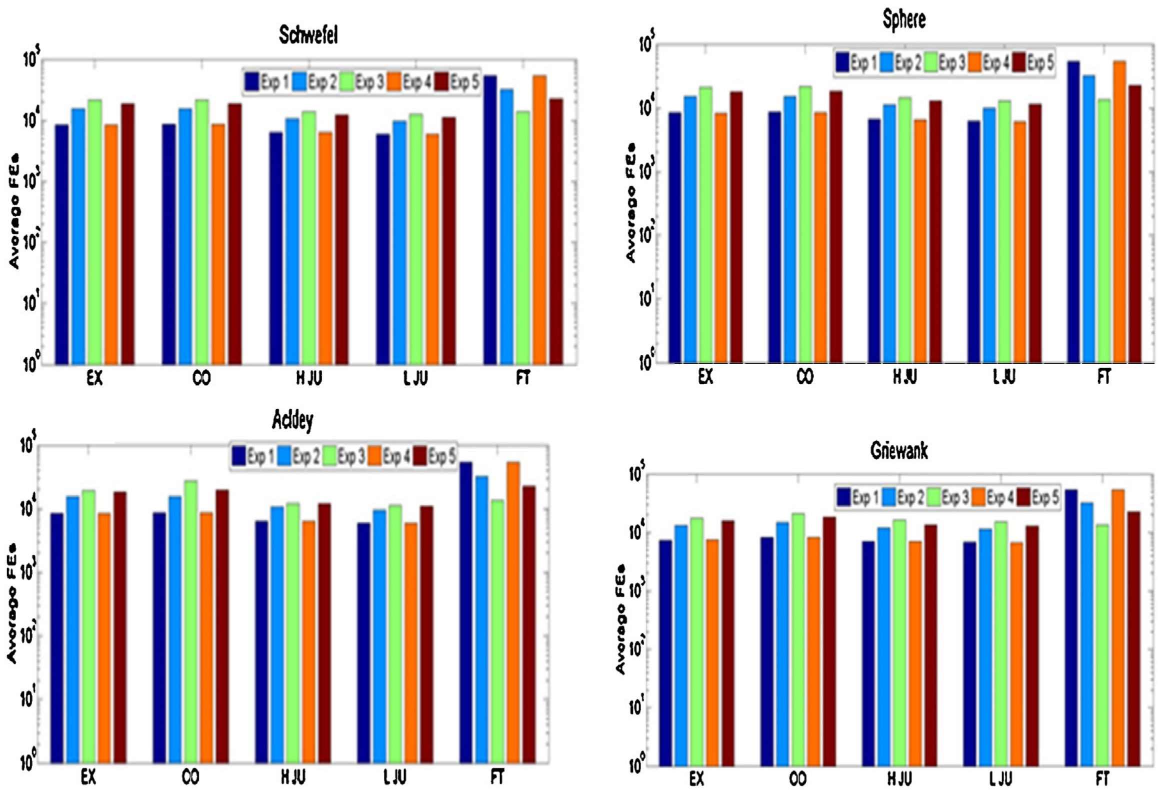


Fig. 14. Average FEs of each RLMPSO operation for (a) Sphere, (b) Schwefel, (c), Ackley (d) Griewank functions.

Table 6
Contribution of each RMPLSO operation.

Function		Without exploration	Without convergence	Without high jump	Without low jump	Without fine-tuning
Sphere	*Mean*	6.37e−49	1.89e−40	1.53e−48	2.15e−45	46.64
	Std	(1.82e−48)	(4.02e−40)	(4.85e−48)	(5.67e−45)	(45.26)
Schwefel	*Mean*	9.03e−29	1.15e−24	2.37e−29	5.64e−28	0.64
	Std	(2.36e−25)	(2.42e−24)	(6.67e−28)	(1.39e−27)	(0.53)
Ackley	*Mean*	4.53e−14	2.16e−12	1.57e−14	5.99e−14	3.65
	Std	(1.02e−14)	(4.69e−12)	(4.81e−14)	(1.85e−14)	(0.44)
Griewank	*Mean*	2.50e−05	0.0039	5.25e−05	5.25e−04	0.0019
	Std	(8.71e−04)	(0.0031)	(3.57e−05)	(1.16e−02)	(0.0031)

operations omitted. As can be seen in Table 6, the most important operations affecting RLMPSO were convergence and fine-tuning. This was because of the nature of the benchmark problems, i.e. small numbers of local optima [1]. The least important operations were exploration and high jump. However, both operations would be useful for complicated benchmark problems with high numbers of local optima, such as the composite benchmark problems in Case Study II.

4.2.4. Analysis of the RLMPSO behaviour at run-time

To trace the sequence of each RLMPSO operation at run time, Table 7 shows the detailed information of each particle as well as the identification (denoted as ID) of the global best particle. The following abbreviations are used in the illustration:

- (H) Particle executed the high jump operation, and improved the local best value.
- (h) Particle executed the high jump operation, and could not improve the local best value.
- (−) Particle was not selected.
- (L) Particle executed the low jump operation, and improved the local best value.
- (l) Particle executed the low jump operation, and could not improve the local best value.
- (C) Particle executed the convergence operation, and improved the local best value.
- (c) Particle executed the convergence operation, and could not improve the local best value.
- (E) Particle executed the exploration operation, and improved the local best value.
- (e) Particle executed the exploration operation, and could not improve the local best value.
- (F) Particle executed the fine-tuning operation, and improved the local best value.
- (f) Particle executed the fine-tuning operation, and could not improve the local best value.

As an example, at the fifth FEs, particle 2 successfully executed the convergence operation, and became the global best particle. On the other hand, particle 3 successfully executed the exploration operation at FEs = 9. While its local best particle was updated, it was

Table 7
Analysis of the particle execution sequence.

Particle 1	- - - h - - E - - E - - e - - e - - e - - C - - C - - ..... - -
Particle 2	- - - - C - - e - - C - - l - - l - - c - - c - - C - ..... - -
Particle 3	- - - - - e - - E - - l - C - e - - e - - e - - c ..... f F
Global ID	3 3 3 3 2 2 2 2 2 1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 1 1 1 ..... 3 3
FEs	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 ..... 1352 1353

Table 8
Control parameters of RLMPSO.

Parameter	Number of setting levels		
N (population size)	−1 (3)	0 (5)	+1 (10)
D (minimum lapse of RLMPSO FEs)	−1 (10)	0 (100)	+1 (1000)
C (cost of local search operation)	−1 (−2)	0 (−4)	+1 (−8)
J (number of fine-tuning FEs)	−1 (5)	0 (30)	+1 (100)
V (the range of velocity $[V_{min}, V_{max}]$)	−1 (0.2)	0 (0.5)	+1 (0.8)

inferior to that of particle 2. However, particle 3 was able to achieve the best results at FEs = 14, and became the global best particle.

In addition, it can be concluded from Table 7 that when a particle had executed an operation successfully, the operation was accorded a higher priority to be executed in the next FEs. Notice that the fine-tuning operation was delayed until a minimum lapse of 1000 FEs, as shown in Table 7.

4.2.5. Sensitivity analysis of the RLMPSO control parameters

To investigate the effect of the key RLMPSO control parameters, i.e., N (population size), D (minimum lapse of RLMPSO FEs before executing fine-tuning), C (cost of the local search operation), J (number of fine-tuning FEs), and V (the range of velocity), a graphical sensitivity technique as used in [72] was followed. The main idea was to measure the influence of each parameter independently with respect to the RLMPSO performance. The effects of five RLMPSO control parameters are shown in Table 8. During the experiment, the remaining RLMPSO parameters were set to those in PSO and ISPO models recommended in the literature, i.e., [67], [20,49–51] and [12] as explained in Sections 3.5, 3.6, and 3.7.

During this experiment, each parameter was independently changed from low (i.e. −1) to high (i.e. +1) as shown in Table 8, and the settings of the remaining parameters followed the suggested values in Table 1. As an example, when parameter N was studied, other parameters (i.e. D, C, J, and V) were set according to Table 1. For each parameter analysis, RLMPSO was executed 100 times for each of the three levels (i.e. −1, 0, and +1). After that, the mean fitness value was computed, as shown in Fig. 15. The main benefit of the sensitivity analysis is to show the change of each parameter graphically, i.e., having an increasing effect, a decreasing effect, or no effect with respect to the RLMPSO performance.

From the graphical plot in Fig. 15, it can be seen that the most important parameter that affected the performance of RLMPSO in all benchmark problems was J (the fine-tuning FEs). By increasing J, the number of FEs calls consumed by the fine-tuning operation is increased. As can be seen in Fig. 15, when J was high (i.e. at state +1 with 100 FEs), the RLMPSO performance for the unimodal benchmark problems (i.e. Sphere and Schwefel) improved, but became worse for the multimodal benchmark problems (i.e. Ackely and Griewank). Because of many local optima in multimodal problems, the fine-tuning operation was less effective, as compared with the unimodal problems with only a single global solution. The least important factor was V (the range of velocity) in all benchmark problems, as shown in Fig. 15.

A further analysis has been carried on by evaluating the effect and relative effect measures as defined in [72]:

$$\textit{relative effect} = 100 \times \frac{\textit{effect}}{\textit{global mean}} \tag{13}$$

$$\begin{aligned} \textit{effect} = \max \Big(& abs\left(\overline{\log(f(x))_0} - \overline{\log(f(x))_{+1}}\right), \\ & abs\left(\overline{\log(f(x))_0} - \overline{\log(f(x))_{-1}}\right), \\ & abs\left(\overline{\log(f(x))_{+1}} - \overline{\log(f(x))_{-1}}\right)\Big) \end{aligned} \tag{14}$$

where the global mean is the mean effect among all parameters, $\overline{\log(f(x))_0}$ is the log mean fitness function at parameter setting (0), $\overline{\log(f(x))_+}$ is the log mean fitness function at parameter setting (+1), and $\overline{\log(f(x))_-}$ is the log mean fitness function at parameter setting (−1).

The effect and relative effect measures are reported in Table 9. Parameters C (the cost of local search) and J (the local search FEs) showed the highest impact on the RLMPSO performance, as compared with those from other parameters. This implied that managing the local search method efficiently constituted one of the most important issues in developing an effective memetic-based algorithm.

4.2.6. RLMPSO diversity analysis

An analysis of the diversity curve generated by RLMPSO during the execution time of a 2-D sphere optimization function was conducted. The 2-D sphere optimization function is defined as:

$$\textit{Minimize}\, f(x) = x_1^2 + x_2^2, \quad x_i \in [-10, 10] \tag{15}$$

Following [69], the diversity analysis measure is defined as:

$$\textit{Diversity}(t) = \frac{1}{N|L|} \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{D} \left(\boldsymbol{X}_i^j - \overline{\boldsymbol{X}^j} \right)^2} \tag{16}$$

where t is the current search FEs, N is the total number of particles, L is the longest diagonal length in the search space, D is the dimension of the search space, $\boldsymbol{X}_i^j$ is the value of particle i at dimension j, and $\overline{\boldsymbol{X}^j}$ is the mean value of the whole swarm particles at dimension j.

For comparison purpose, PSO [61] was employed with a total of 30 particles, and the maximum number of FEs was set to 1000. Fig. 16 shows the diversity measures for both PSO and RLMPSO. The RLMPSO diversity curve moved up and down during the search operation. This was owing to the dynamic behaviour of RLMPSO, where each particle evolved independently and could execute any search operations under the control of RL. However, in general, the diversity curve decreased from high to low as the search process progressed.

Further analyses were carried on by plotting the locations of the particles at four different search FEs (i.e. FEs = 1, 200, 500, and

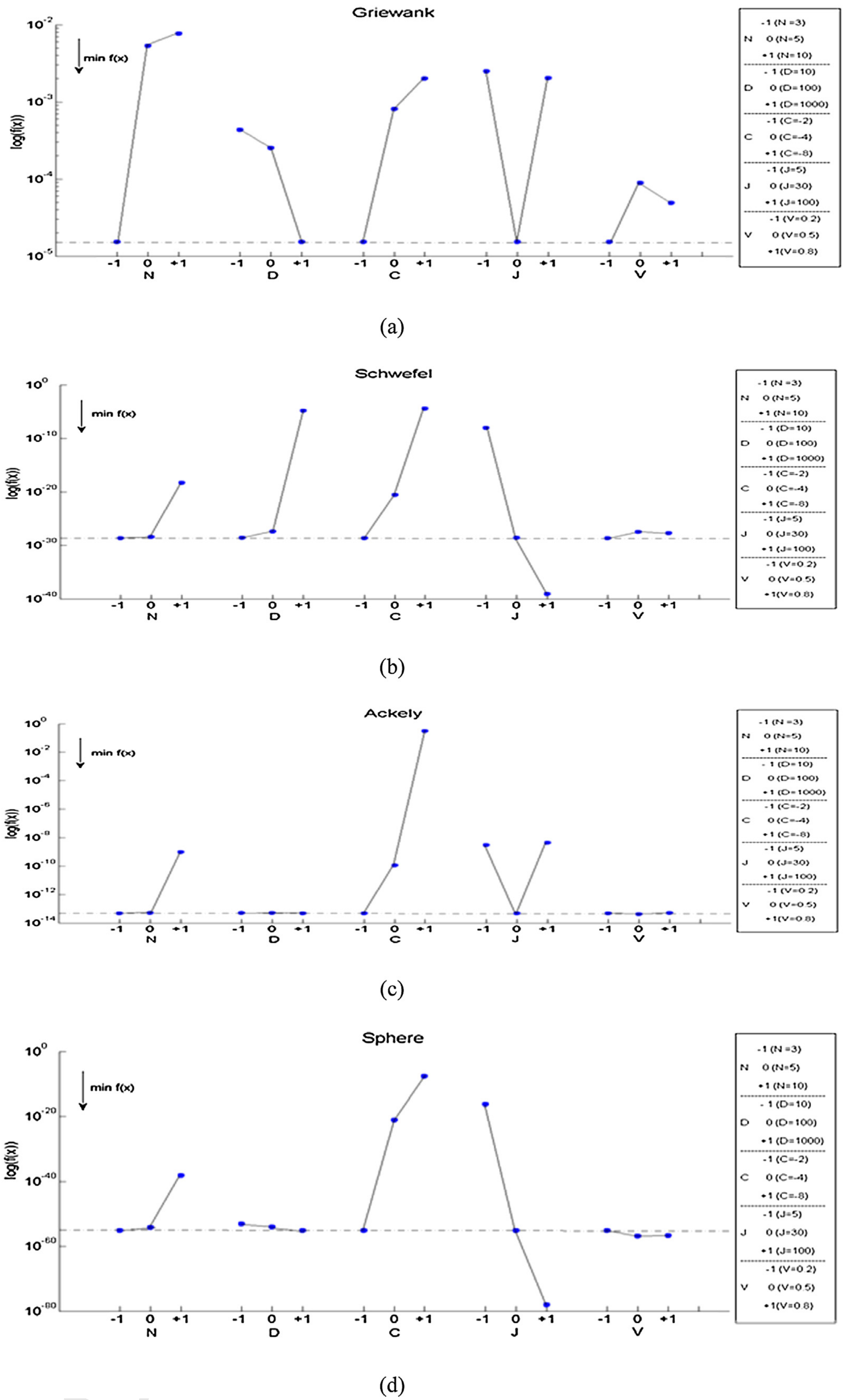


Fig. 15. (a)–(d) Sensitivity analysis of the RLMPSO control parameters.

Table 9
The effect and relative effect measure on RLMPSO performances.

		N	*D*	*C*	*J*	*V*
Sphere	Effect	6.74e−39	7.30e−54	2.36e−08	4.93e−17	6.45e−56
	Relative effect	1.4280e−28	1.5466e−43	500.00	1.05e−06	1.37e−45
Schwefel	Effect	4.45e−19	1.54e−05	4.02e−05	9.97e−09	1.5750e−28
	Relative effect	4.00e−12	138.46	361.45	0.09	1.42e−21
Ackley	Effect	9.54e−10	3.30e−15	0.31e−00	4.45e−09	3.10e−15
	Relative effect	1.54e−06	5.32e−12	500.00	7.18e−06	5.00e−12
Griewank	Effect	0.10 e−01	4.17e−04	0.20 e−02	0.25 e−02	7.33e−05
	Relative effect	333.55	13.91	66.71	83.39	2.44

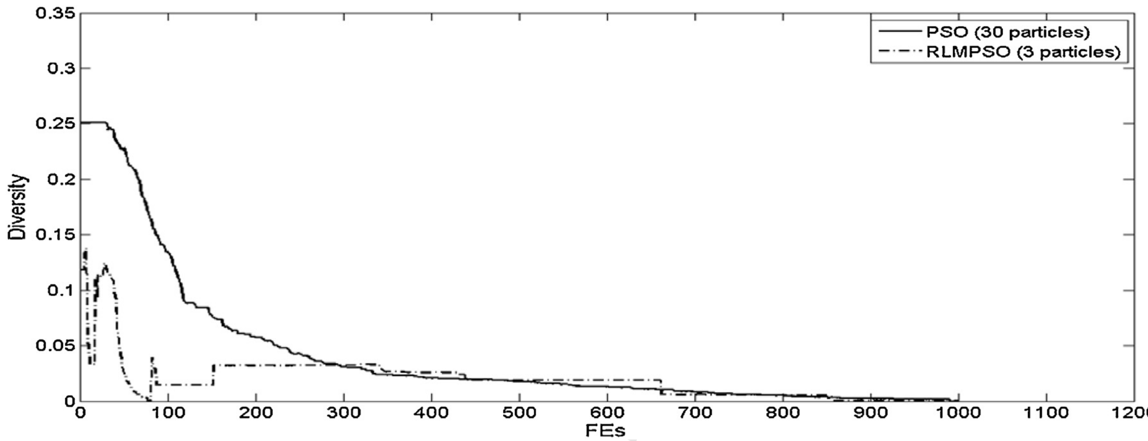


Fig. 16. Diversity curve of RLMPSO and PSO algorithm.

900), as shown in Fig. 17. The particles started with the exploration operation at the beginning of the search process and then shifted gradually to the convergence state, i.e., the swarm particles became crowded around the global best particle.

4.3. Comparison with other PSO variants

Table 10 reports the average fitness values from 100 runs of RLMPSO. For comparison purposes, the reported results in [1,68,69] are included in Table 10. Note that the results in Table 10 were generated using the same number of FEs in [1], i.e. $FE_{max} = 2.5 \times 10^5$, and the configuration for all benchmark problems were set at 30-D, and each experiment was repeated 100 times as in [1]. In [68,69], the benchmark problems were set at the same dimension (i.e. 30-D) while the experimental runs were 30 times, and the FEs in [68,69] were 3×10^5 and 2×10^5, respectively.

Therefore, the maximum FEs of RLMPSO (i.e., $FE_{max} = 2.5 \times 10^5$) was the same as that in [1] (i.e., $FE_{max} = 2.5 \times 10^5$), lower than that in [68] (i.e., $FE_{max} = 3 \times 10^5$), but higher than that in [69] (i.e., $FE_{max} = 2 \times 10^5$).

As can be seen in Table 10, RLMPSO outperformed the memetic-based PSO variant in [1] for all four benchmark problems and the methods in [68] and [69] for three benchmark problems. However, RLMPSO yielded inferior results than those reported in [68,69] for the Ackley function. In addition, it should be noted that since the method in [69] was executed with fewer number of FEs as compared with that of RLMPSO, it could outperform RLMPSO if it was executed with $FE_{max} = 2.5 \times 10^5$. As such, it was not surprising that the result from the method in [69] was better than that in [68] for the Ackley function, since more FEs were consumed [73].

To quantify the achieved results statistically, the 95% confidence intervals of the RLMPSO results were computed using the bootstrap method [74], as shown in parentheses in Table 10. Statistically, RLMPSO significantly outperformed other methods, except the Ackley benchmark problem. The refinement capability of RLMPSO allowed it to outperform other methods studied in this comparison.

4.4. Case Study II: composite benchmark problems

In this case study, six composite functions in [1] were examined. They constituted more challenging benchmark problems as compared with the unimodal and multi-modal functions in Case Study I. As an example, composite function five (*cf5*) is composed of ten benchmark functions comprising two rotated Rastrigin functions, two rotated Weierstrass functions, two rotated Griewank functions, two rotated Ackley functions, and two sphere functions.

The same benchmark composite problems were studied in [1] with three PSO variants i.e., CLPSO [53], ISPO [12], and POMA

Table 10
Comparison between the RLMPSO results and other reported results in the literature.

Function	[68] Mean	[69] Mean	[1] Mean	RLMPSO Mean (95% confidence interval)
Sphere	2.78e−49	1.35e−30	1.04e−20	**6.62e−56 (1.74e−55, 7.42e−57)**
Schwefel	1.35e−26	–	4.08e−10	**2.25e−29 (1.68e−29, 2.81e−29)**
Ackley	3.47e−14	**1.69e−14**	0.415	4.81e−14 (4.49e−14, 5.13e−14)
Griewank	2.06e−0	2.54e−2	1.807e−3	**1.52e−05 (0, 1.54e−05)**

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

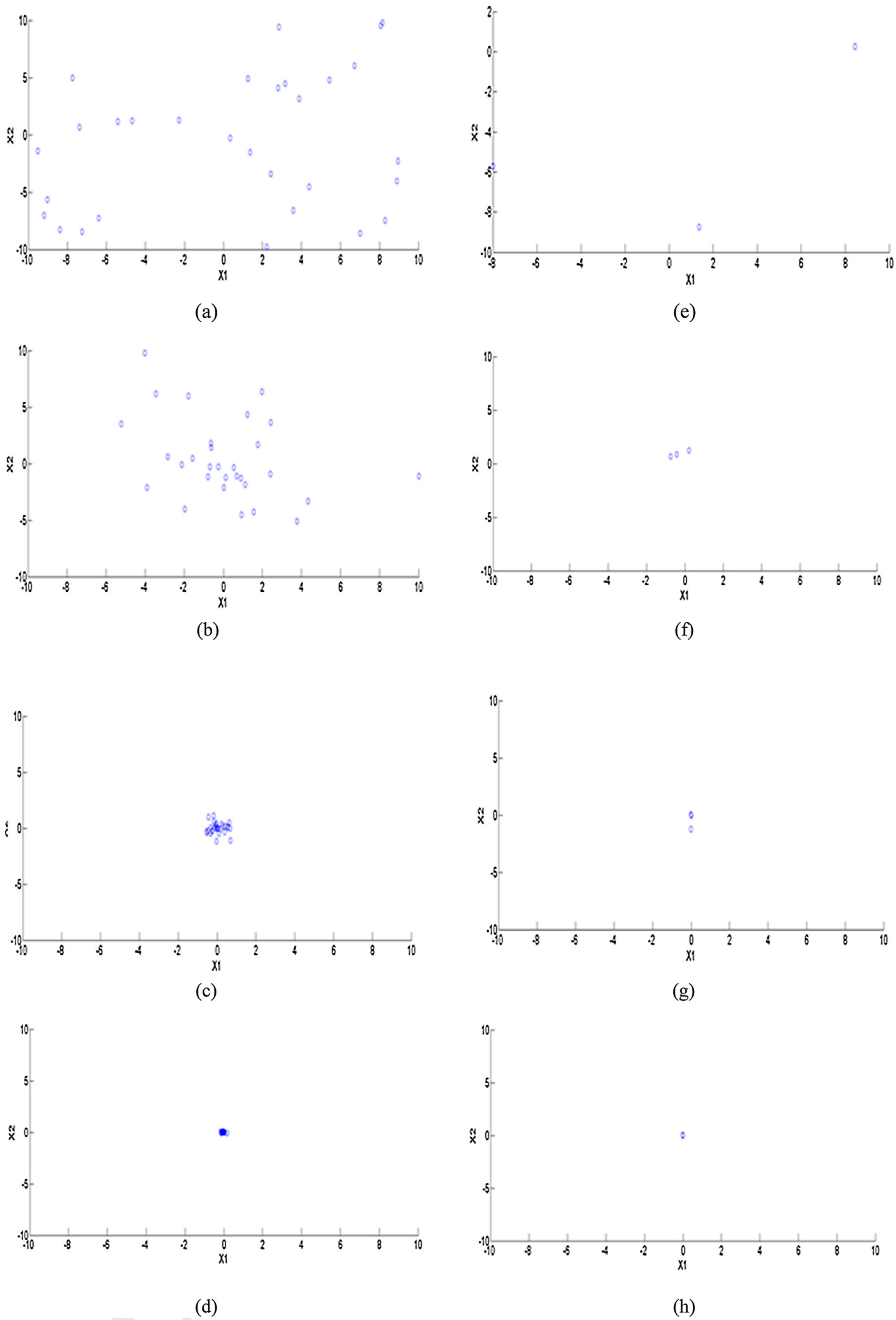


Fig. 17. Population distributions of RLMPSO as compared with PSO, i.e., (a)–(d) distribution of PSO at FEs = 1, 200, 500, and 900; (e)–(h) distribution of RLM PSO at FEs = 1, 200, 500, and 900, respectively.

Table 11
Results for the composite problems.

Function	CLPSO [53] Mean	ISPO [12] Mean	POMA [1] Mean	RLMPSO Mean (95% confidence interval)
cf1	45.07	252.00	8.00	**1.20e−01 (3.48 e−01, 6.5000e−04)**
cf2	89.86	362.01	47.29	**27.0757 (20.7540, 33.3975)**
cf3	201.06	480.0	**148.77**	157.02 (143.96, 170.34)
cf4	356.04	671.23	377.85	**320.91 (312.26, 329.96)**
cf5	62.49	435.99	39.74	**23.47 (9.23, 48.94)**
cf6	742.58	851.96	673.80	**495.21 (475.77, 505.74)**

[1]. The results from [1] are included in Table 11 for comparison purposes. The maximum number of FEs was set to $FE_{max} = 2.5 \times 10^5$, as used in [1], and the dimension of all benchmark problems was 10-D. The experiment was repeated 100 times. The mean fitness values are shown in Table 11.

RLMPSO yielded the best results as compared with those from other PSO variants, except POMA in cf3. One of the reasons pertaining to the good performance of RLMPSO was because of fine-tuning, whereby each particle in the micro swarm had the chance to undergo the refinement operation. In addition, the capability of each particle to change from convergence to exploration at any time provided RLMPSO a better chance to escape from local optima. As a result, RLMPSO outperformed other methods in most of the benchmark problems.

The 95% confidence intervals are shown in parentheses in Table 11. Note that the upper limit of the 95% bootstrapped confidence interval is smaller than the reported mean fitness values of the related methods in all functions, except for cf3, whereby the POMA result resides within the 95% confidence interval of RLMPSO.

4.5. Case Study IIII: shifted and rotated benchmark problems

To investigate the effectives of RLMPSO in solving shifted and rotated benchmark problems, a total of ten functions from CEC 2005 [75] were used. These problems have been widely studied in the literature [50,68,76,77]. Four models were evaluated using the same CEC 2005 functions, i.e., CLPSO [53], CPSO [31], ANS [78], and GWO [79]. The settings of these models are shown in Table 12. The mathematical formulae of the employed benchmark functions are defined, as follows [75].

F1: Shifted sphere function

$$F_1(x) = \sum_{i=1}^{D} \mathbf{Z}_i^2 + f\_bias, \ \mathbf{Z} = \mathbf{X\text{-}O}, \ \mathbf{X} = [x_1, x_2, \ldots, x_D]$$

D: dimensions, $X \in [-100, 100]^D$
O: the shifted global optima $\mathbf{O} = [o_1, o_2, \ldots, o_D]$

Table 12
parameter settings of RLMPSO and other algorithms.

Algorithm	Dimension	Population size	Parameter settings
GWO	30	30	a: 2 –1
CPSO	30	30	$c_1 = c_2 = 1.49$, w: 0.9–0.5, group number equals to dimensionality
CLPSO	30	30	w: 0.9–0.4, $c_1 = c_2 = 2$, $m = 7$
ANS	30	30	$\sigma = 0.5$, and $n = 10$
RLMSO	30	3	The same settings in Table 1

f_bias: the bias value
F2: Shifted Schwefel's Problem 1.2

$$F_2(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} Z_j \right)^2 + f\_bias, \quad \mathbf{Z} = \mathbf{X\text{-}O},$$

$\mathbf{X} = [x_1, x_2, \ldots, x_D]$

D: dimensions, $X \in [-100, 100]^D$
O: the shifted global optima $\mathbf{O} = [o_1, o_2, \ldots, o_D]$
F3: Shifted Schwefel's Problem 1.2 with noise in fitness

$$F_4(x) = \left(\sum_{i=1}^{D} \left(\sum_{j=1}^{i} Z_j \right)^2 \right) * (1 + 0.4 \left| N(0,1) \right|) + f\_bias, \quad \mathbf{Z} =$$

$\mathbf{X\text{-}O}$,
$\mathbf{X} = [x_1, x_2, \ldots, x_D]$
D: dimensions, $X \in [-100, 100]^D$
O: the shifted global optima $\mathbf{O} = [o_1, o_2, \ldots, o_D]$
F4: Shifted rotated Weierstrass function

$$F_9(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{\max}} \left[a^k \cos\left(2\pi b^k (Z_i + 0.5) \right) \right] \right) -$$

$$D \sum_{k=0}^{k_{\max}} \left[a^k \cos\left(2\pi b^k (0.5) \right) \right] + f\_bias$$

$a = 0.5$, $b = 3$, $a = 0.5$, $b = 3$, $k_{max} = 20$
$\mathbf{Z} = (\mathbf{X\text{-}O})^*\mathbf{M}$, $\mathbf{X} = [x_1, x_2, \ldots, x_D]$
D: dimensions, $X \in [-5, 5]^D$
O: the shifted global optima $\mathbf{O} = [o_1, o_2, \ldots, o_D]$
M: linear transformation matrix for function rotation
F5: Shifted Rastrigin's Function

$$F_8(x) = \sum_{i=1}^{D} ({Z_i}^2 - 10\cos(2\pi Z_i) + 10) + f\_bias$$

$\mathbf{Z} = (\mathbf{X\text{-}O})$, $\mathbf{X} = [x_1, x_2, \ldots, x_D]$
D: dimensions, $X \in [-5, 5]^D$
O: the shifted global optima $\mathbf{O} = [o_1, o_2, \ldots, o_D]$

4.5.1. Mean fitness value analysis

The mean fitness values achieved by RLMPSO and other methods for the rotated and shifted benchmark problems are shown in Table 13. Each method was executed 30 times with a total of 30×10^4 *FEs* at 30-D using the parameter settings in Table 12. It can be seen in Table 13 that RLMPSO compared favourably with other methods. In particular, RLMPSO achieved the highest accuracy scores for F1, F2, and F5.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

Table 13
The mean fitness values for the rotated and shifted functions.

Algorithm	F1	F2	F3	F4	F5
GWO	9.66E−05 ± 7.71E−05	1.06E+04 ± 9.14E+02	1.22E+04 ± 5.61E+03	1.25E+02 ± 3.94E+00	1.03E−04 ± 7.83E−05
CPSO	7.29E−05 ± 3.19E−05	0.57E−02 ± 0.50E−02	2.40E+04 ± 7.48E+03	**1.07E+02 ± 4.43E+00**	1.58E−04 ± 1.15E−04
CLPSO	2.50E−11 ± 4.78E−12	2.04E−12 ± 4.30E−12	4.50E+03 ± 6.14E+02	1.14E+02 ± 2.34E+00	1.98E−12 ± 2.52E−12
ANS	0.28E−02 ± 0.16E−02	0.11E−01 ± 0.21E−02	**1.15E+02 ± 1.26E+02**	1.12E+02 ± 1.06E+00	4.70E−04 ± 1.88E-04
RLMPSO	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**	3.73E+04 ± 7.85E+03	1.21E+02 ± 3.71E+00	**0.00E+00 ± 0.00E+00**

4.5.2. Convergence curve analysis

To investigate the characteristics of RLMPSO at run time for the shifted and rotated benchmark problems, a graphical comparison analysis technique was used by plotting the convergence curves of RLMPSO and other methods. Specifically, the base-10 logarithmic mean values of the fitness function from a total of 30 runs were computed, as shown in Fig. 18. The convergence speed of RLMPSO was slower than those from other models. This was because of the small population size (i.e. 3 particles) of RLMPSO. However, RLMPSO could escape from local optima owing to the benefit of the jumping operations, as well as its capability of changing from exploration to convergence at any time during the search process. As an example, RLMPSO started with a slow convergence rate in Fig. 18(a), (b), and (e), but was able to converge rapidly once the global optima regions were identified.

4.5.3. Computational time analysis

To analyze the computational time, the evaluation criteria in [75] were adopted. The general steps of the criteria are explained, as follows:

Step 1: Run and compute the time consumed by the code segment in Fig. 19. This code segment was suggested in [75] to measure the time required for executing different mathematical operations such as summation division, multiplication operations. The time consumed is represented by variable T_0.

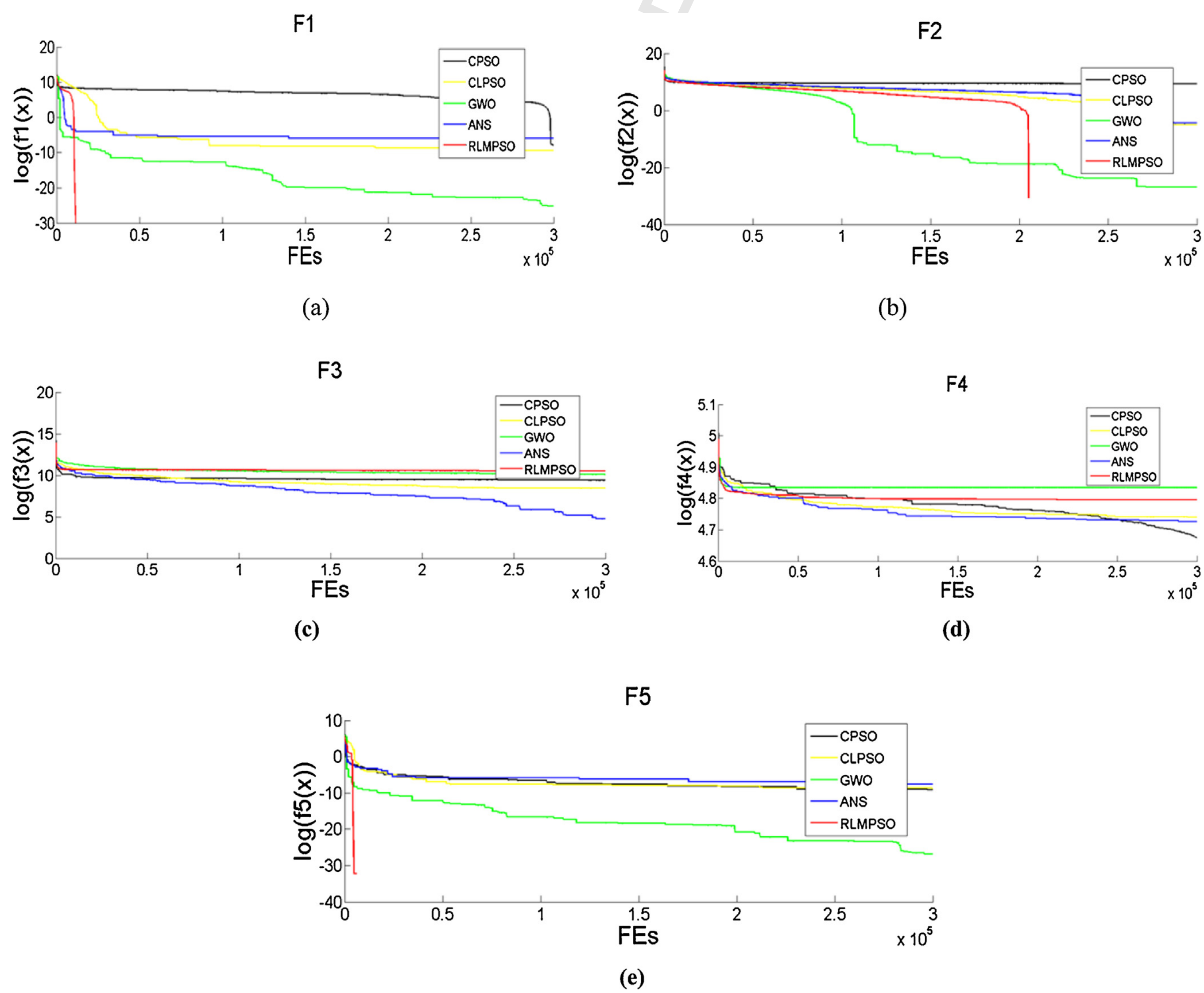


Fig. 18. The convergence curves of RLMPSO and other methods (a) F1, (b) F2, (c) F3, (d) F4, (e) and F5.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

$$\begin{aligned}
&\textbf{for } i = 1{:}100000\\
&\qquad x = double(5.55);\\
&\qquad x = x + x;\, x = \frac{x}{2};\, x = x * x;\\
&\qquad x = sqrt(x);\, x = \ln(x)\,;\, x = \exp(x)\,;\, x = \frac{x}{x}\\
&\textbf{end}
\end{aligned}$$

Fig. 19. The code segment for evaluating T_0.

Table 14
Results of computational complexity in seconds.

Algorithm	T_0	T_1	T_2	$(T_2-T_1)/T_0$
GWO	3.49e−05	12.23	25.67	3.85e+5
CPSO			42.78	8.75e+5
CLPSO			34.54	6.39e+5
ANS			302.1	8.31e+6
RLMPSO			21.89	2.77e+5

Step 2: Compute the time required to evaluate function F3 (i.e. *Shifted Rotated High Conditioned Elliptic Function*) from [75] with a dimension of 50-D for 200,000 FEs according to [75]. The time consumed is represented by variable T_1.

Step 3: Compute the time required by the entire model with function F3 at 50-D for 200,000 FEs according to [75]. This step is conducted independently for each model, i.e. PSO, CLPSO, DE, BAT, Harmony, GWO, and RLMPSO. The time consumed is represented by variable T_2.

Repeat Step 3 for five times and compute the mean of T_2, i.e., $\overline{T_2} = mean(T_2)$. The time complexity is represented by T_0, T_1, T_2, and $(\overline{T_2} - T_1)/T_2$, as indicated in Table 14. The computational time required by each model (i.e. T_2) was similar except RLMPSO which required a slightly shorter time (i.e. 21.89 s). This was owing to the small population size of RLMPSO (i.e. 3 particles), as compared with other methods that worked with a large population size of 30. Additionally, RLMPSO with a large population size (i.e. 30 particles) was experimented, and the results are shown in Table 15. It should be noted that the CPU time consumed by each method is affected by several factors such as programming language and programming skill, as well as hardware configuration.

A detailed analysis was conducted to investigate the time consumed by the Q-table operations of RLMPSO, i.e. updating the Q-table and obtaining the best action from the Q-table. The computational times required for both operations are reported in Table 15. With three particles, the time consumed was very small as compared with the total time consumed by the entire model (i.e. T_2). Note that the Q-table size was small (i.e. 5×5) and was independent from the dimension of the problem. Moreover, a large population size of RLMPSO (i.e. 30 particles) was experimented, with the maximum FE set to 200,000. As can be seen in Table 15, the computational time of RLMPSO increased (i.e. T_2 = 29.87 s) owing to extra memory requirements.

4.5.4. Statistical evaluation measure

The *t*-test [80] was conducted to statistically evaluate the achieved results by RLMPSO as compared with other methods in solving the shifted and rotated functions of CEC 2005. For comparing two methods (X and Y, where X = RLMPSO and Y = the compared method), the null hypothesis H_0 claimed that X and Y performed equally well. The alternative hypothesis H_1 assumed that X outperformed Y. The significance level of *p-value* was set at 0.05, i.e., the alternative hypothesis H_1 would be accepted if the *p-value* was less than 0.05 (i.e., 95% confidence level). Table 16 presents the *p-values* from the paired *t*-test between RLMPSO and other methods. All the *p-values* were smaller than 0.05, except for the test between RMLPSO and CLPSO for F4.

Table 16
The *p*-values of the statistical *t*-test.

Function	F1	F2	F3	F4	F5
GWO	0.0042	0.0000	0.0014	0.0002	0.0235
CPSO	0.0011	0.0389	0.0002	0.0426	0.0195
CLPSO	0.0267	0.0298	0.0043	**0.1675**	0.0115
ANS	0.0025	0.0001	0.0001	0.0027	0.0017

Table 17
The results of the gear design problem.

Algorithm	Mean	95% confidence interval
[51]	5.72E−09	–
[68]	2.22E−09	–
[49]	4.25E−09	–
RLMPSO	**1.6300e−11**	**(2.0833e−11, 9.5000e−12)**

4.6. Case Study IIII: real-world benchmark problems

Two real-world engineering design optimization problems were examined, i.e., train gear design and pressure vessel design, as follows.

4.6.1. Gear design problem

The problem of designing train gears was studied in [81], and was further examined in [49,51,68]. The main objective of the problem was to optimize the gear ratio of a compound train gear containing three gears, as shown in Fig. 20. The optimization problem is defined as follows:

$$f(x) = \left(\frac{1}{6.931} - \frac{AD}{BC}\right)^2 \tag{17}$$

where A, B, C and D are the decision variables that represent the number of gear teeth and their range, i.e., 12 =< A, B, C, and $D \le 60$, as described in [81].

The main objective was to find the optimal values of A, B, C and D that could produce a gear ratio as close to 1/6.931 as possible. The formulation of the gear ratio is as follows:

$$\textit{The gear ratio} = \frac{\textit{angular velocity of output shaft}}{\textit{angular velocity of input shaft}} \tag{18}$$

In this study, the performance of RLMPSO was compared with the reported results in [49,51,68]. The parameters used in this experiment were the same as those in [49,51,68], where the maximum FEs was fixed at $FE_{max} = 3 \times 10^5$. As indicated in Table 17, RLMPSO achieved the best results. Again, the capability of performing fine-tuning was useful to tackle this train gear design optimization problem. Furthermore, from the statistical point

Table 15
Results of the RLMPSO computational time in seconds.

Algorithm	T_0	T_1	T_2	$(T_2-T_1)/T_0$	Q-Table Update operation	Q-Table Get best operation
RLMPSO with 3 particles	3.49e−05	12.23	21.89	2.77e+5	0.12	0.18
RLMPSO with 30 particles			29.87	5.12e+5	0.86	1.2

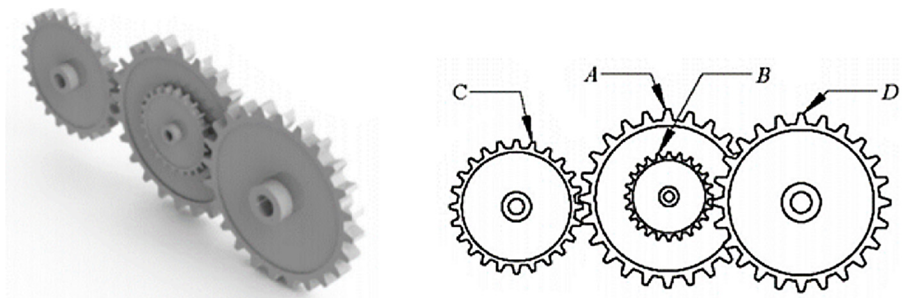


Fig. 20. The gear design problem [82].

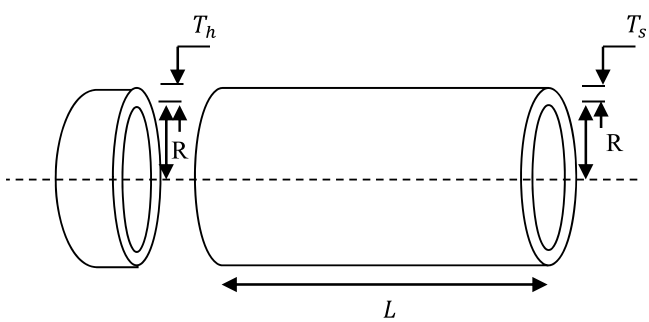


Fig. 21. The pressure vessel design problem.

of view, RLMPSO significantly outperformed other methods in [49,51,68], as indicated by the 95% confidence intervals.

4.6.2. Pressure vessel design problem

The pressure vessel design problem [83–85] aimed to find the minimum manufacturing cost of designing the cylindrical compressed air storage with pre-defined conditions and constrains, as shown in Fig. 21. The complexity of this design problem was higher than that of the train gear design problem as a total of four design constraints (i.e. g_1, g_2, g_3, and g_4) were involved. The problem is defined as follows.

$$\textit{Consider}\ \vec{x} = [T_s, T_h, R, L]$$

$$\textit{Minimize}\, f(\vec{x}) = 0.6224\,T_s RL + 1.7781\,T_h R^2 + 3.1661 T_s^2\, L + 19.84\, T_s^2\, R \tag{19}$$

$$\textit{subject to}\, g_1(\vec{x}) = 0.0193R - T_s \le 0$$

$$g_2\left(\vec{x}\right) = 0.00954R - T_h \le 0$$

$$g_3\left(\vec{x}\right) = 1296000 - \pi R^2 L - \frac{4}{3}\pi R^3 \le 0$$

$$g_4\left(\vec{x}\right) = L - 240 \le 0$$

where $0 \le T_s \le 99$, $0 \le T_h \le 99$, $10 \le R \le 200$, $10 \le L \le 200$, and L is the length of the cylinder, R is the cylinder radius, T_s is the cylinder thickness, and T_h is the thickness of cylinder head, as shown in Fig. 21.

This experiment was conducted using the same settings in [85], where the maximum FEs was set to $FE_{max} = 5 \times 10^4$, and the experimental run was repeated 30 times. The reported results in [83–85] are shown in Table 18 for comparison purposes. RLMPSO yielded the best mean results as compared with those from other methods. From the statistical point of view, RLMPSO significantly outperformed the methods in [83,84], as indicated by the 95% confidence intervals.

Table 18
Results of the pressure vessel design problem.

Algorithm	Mean	95% confidence interval
[85]	6064.34	–
[84]	6447.74	–
[83]	6410.09	–
RLMPSO	**6028.50**	(5.9577, 6.1251)

5. Summary

A new RLMPSO model has been presented in this paper. RLMPSO operates with a micro population size, with only three particles. It has five dedicated operations, i.e. exploration, convergence, high-jump, low-jump, and fine-tuning. Each particle is able to switch from one operation to another under the control of the RL algorithm. The effectiveness of RLMPSO has been evaluated using four unimodal and multi-modal benchmark problems, six composite benchmark problems, five shifted and rotated benchmark problems, as well as two real-world design problems. The bootstrap confidence intervals as well as the statistical t-test have been used to quantify the performance indicators. From the statistical analysis of the results, the proposed RLMPSO model significantly outperforms a number of PSO variants reported in the literature.

There are a number of areas to be pursued as further work. Firstly, the fine-tuning operation plays a vital role. As such, different local search methods can be incorporated into RLMPSO, such as tabu search [86], simulated annealing [87], and reactive search optimizer [88]. Secondly, the RL algorithm can be used to manage different swarm optimization algorithms such as CLPSO [53], GWO [79], Bee Colony [89], and Harmony [90]. Thirdly, the proposed RLMPSO model can be applied to different real-world optimization problems such as DNA sequence compression [1], flow shop scheduling [2], multi-robot path planning [3], wireless sensor networks [4], and finance applications [5]. Finally, RLMPSO can be used to design SVM-based pattern recognition model [10] by performing simultaneous features selection, parameters tuning, and training instances selection.

References

[1] Z. Zhu, J. Zhou, Z. Ji, Y.-h. Shi, DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm, IEEE Trans. Evol. Computat. 15 (2011) 643–658.
[2] B. Liu, L. Wang, Y.-H. Jin, An effective PSO-based memetic algorithm for flow shop scheduling, IEEE Trans Syst. Man Cybernet. – Part B: Cybernetics 37 (2007) 18–27.
[3] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L.C. Jain, A.K. Nagar, Realization of an adaptive memetic algorithm using differential evolution and q-learning: a case study in multirobot path planning, IEEE Trans. Syst. Man Cybernet.: Syst. 43 (2013) 814–831.
[4] K. Pandremmenou, L.P. Kondi, K.E. Parsopoulos, A study on visual sensor network cross-layer resource allocation using quality-based criteria and meta-heuristic optimization algorithms, Appl. Soft Comput. 26 (2015) 149–165.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

[5] S.C. Chiam, K.C. Tan, A.A. Mamun, A memetic model of evolutionary PSO for computational finance applications, Expert Syst. Appl. 36 (2009) 3695–3711.
[6] L. Jiao, M. Gong, S. Wang, B. Hou, Z. Zheng, Q. Wu, Natural and remote sensing image segmentation using memetic computing, Computat. Intell. Mag. IEEE 5 (2010) 78–91.
[7] S.-H. Yang, J.-F. Kiang, Optimization of asymmetrical difference pattern with memetic algorithm, IEEE Trans. Antennas Propag. 62 (2014) 2297–2302.
[8] Y. Peng, B.-L. Lu, A hierarchical particle swarm optimizer with latin sampling based memetic algorithm for numerical optimization, Appl. Soft Comput. 13 (2013) 2823–2836.
[9] H. Wang, I. Moon, S. Yang, D. Wang, A memetic particle swarm optimization algorithm for multimodal optimization problems, Inform. Sci. 197 (2012) 38–52.
[10] Y. Bao, Z. Hu, T. Xiong, A PSO and pattern search based memetic algorithm for SVMs parameters optimization, Neurocomputing 117 (2013) 98–106.
[11] M. Aziz, M.-H. Tayarani-N, An adaptive memetic Particle Swarm Optimization algorithm for finding large-scale Latin hypercube designs, Eng. Appl. Artif. Intell. 36 (2014) 222–237.
[12] Z. Ji, H. Liao, Y. Wang, Q.H. Wu, A novel intelligent particle optimizer for global optimization of multimodal functions, in: IEEE Congress on Evolutionary Computation, 2007. CEC 2007, 2007, pp. 3272–3275.
[13] D. Tang, Y. Cai, J. Zhao, Y. Xue, A quantum-behaved particle swarm optimization with memetic algorithm and memory for continuous non-linear large scale problems, Inform. Sci. 289 (2014) 162–189.
[14] W.K. Mashwani, A. Salhi, Multiobjective memetic algorithm based on decomposition, Appl. Soft Comput. 21 (2014) 221–243.
[15] C.-L. Chan, C.-L. Chen, A cautious PSO with conditional random, Expert Syst. Appl. 42 (2015) 4120–4125.
[16] Y. Zhang, S. Wang, P. Phillips, G. Ji, Binary PSO with mutation operator for feature selection using decision tree applied to spam detection, Knowledge-Based Syst. 64 (2014) 22–31.
[17] T. Huang, A.S. Mohan, Micro-particle swarm optimizer for solving high dimensional optimization problems (μPSO for high dimensional optimization problems), Appl. Math. Computat. 181 (2006) 1148–1154.
[18] T. Huang, A.S. Mohan, A microparticle swarm optimizer for the reconstruction of microwave images, IEEE Trans. Antennas Propag. 55 (2007) 568–576.
[19] G.S. Piperagkas, G. Georgoulas, K.E. Parsopoulos, C.D. Stylios, A.C. Likas, Integrating particle swarm optimization with reinforcement learning in noisy problems, in: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, ACM, Philadelphia, Pennsylvania, USA, 2012.
[20] Z. Zhi-Hui, Z. Jun, L. Yun, H.S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 39 (2009) 1362–1381.
[21] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Inform. Sci. 181 (2011) 4515–4538.
[22] R. Mallipeddi, S. Mallipeddi, P.N. Suganthan, Ensemble strategies with adaptive evolutionary programming, Inform. Sci. 180 (2010) 1571–1581.
[23] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2011) 1679–1696.
[24] Z. Shi-Zheng, P.N. Suganthan, Z. Qingfu, Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Trans. Evol. Computat. 16 (2012) 442–446.
[25] H. Dong, J. He, H. Huang, W. Hou, Evolutionary programming using a mixed mutation strategy, Inform. Sci. 177 (2007) 312–327.
[26] G. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H. Chen, Differential evolution with multi-population based ensemble of mutation strategies, Inform. Sci. 329 (2016) 329–345.
[27] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings, IEEE International Conference on Neural Networks, 1995, vol. 1944, 1995, pp. 1942–1948.
[28] M. El-Abd, H. Hassan, M. Anis, M.S. Kamel, M. Elmasry, Discrete cooperative particle swarm optimization for FPGA placement, Appl. Soft Comput. 10 (2010) 284–295.
[29] Y. Ren, Y. Wu, An efficient algorithm for high-dimensional function optimization, Soft Comput. 17 (2013) 995–1004.
[30] L. Sun, S. Yoshida, X. Cheng, Y. Liang, A cooperative particle swarm optimizer with statistical variable interdependence learning, Inform. Sci. 186 (2012) 20–39.
[31] F. Van den Bergh, A.P. Engelbrecht, A Cooperative approach to particle swarm optimization, IEEE Trans. Evol. Computat. 8 (2004) 225–239.
[32] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Computat. 16 (2012) 210–224.
[33] F. Zhao, G. Li, C. Yang, A. Abraham, H. Liu, A human–computer cooperative particle swarm optimization based immune algorithm for layout design, Neurocomputing 132 (2014) 68–78.
[34] S. Duman, N. Yorukeren, I.H. Altas, A novel modified hybrid PSOGSA based on fuzzy logic for non-convex economic dispatch problem with valve-point effect, Int. J. Elect. Power Energy Syst. 64 (2015) 121–135.
[35] A. Gálvez, A. Iglesias, A new iterative mutually coupled hybrid GA–PSO approach for curve fitting in manufacturing, Appl. Soft Comput. 13 (2013) 1491–1504.
[36] M.S. Kiran, M. Gündüz, A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems, Appl. Soft Comput. 13 (2013) 2188–2203.
[37] R. Rahmani, R. Yusof, M. Seyedmahmoudian, S. Mekhilef, Hybrid technique of ant colony and particle swarm optimization for short term wind energy forecasting, J. Wind Eng. Ind. Aerodyn. 123 (Part A) (2013) 163–170.
[38] R. Pandi, B.K. Panigrahi, Dynamic economic load dispatch using hybrid swarm intelligence based harmony search algorithm, Expert Syst. Appl. 38 (2011) 8509–8514.
[39] S.Z. Zhao, P.N. Suganthan, Q.-K. Pan, M. Fatih Tasgetiren, Dynamic multi-swarm particle swarm optimizer with harmony search, Expert Syst. Appl. 38 (2011) 3735–3742.
[40] L. Ma, M. Gong, J. Liu, Q. Cai, L. Jiao, Multi-level learning based memetic algorithm for community detection, Appl. Soft Comput. 19 (2014) 121–133.
[41] T.K. Das, G.K. Venayagamoorthy, U.O. Aliyu, Bio-inspired algorithms for the design of multiple optimal power system stabilizers: SPPSO and BFA, IEEE Trans. Industry Appl. 44 (2008) 1445–1457.
[42] V.H. Hinojosa, R. Araya, Modeling a mixed-integer-binary small-population evolutionary particle swarm algorithm for solving the optimal power flow problem in electric power systems, Appl. Soft Comput. 13 (2013) 3839–3852.
[43] K.E. Parsopoulos, Parallel cooperative micro-particle swarm optimization: a master–slave model, Appl. Soft Comput. 12 (2012) 3552–3579.
[44] M.A.M. de Oca, T. Stutzle, M. Birattari, M. Dorigo, Frankenstein's PSO: a composite particle swarm optimization algorithm, IEEE Trans. Evol. Computat. 13 (2009) 1120–1132.
[45] M. Hu, T. W, J.D. W, An adaptive particle swarm optimization with multiple adaptive methods, IEEE Trans. Evol. Computat. 17 (2013) 705–720.
[46] S. Sun, J. Li, A two-swarm cooperative particle swarms optimization, Swarm Evol. Computat. 15 (2014) 1–18.
[47] X. Li, Niching without niching parameters: particle swarm optimization using a ring topology, IEEE Trans. Evol. Computat. 14 (2010) 150–169.
[48] W. Zhang, D. Ma, J.-j. Wei, H.-f. Liang, A parameter selection strategy for particle swarm optimization based on particle positions, Expert Syst. Appl. 41 (2014) 3576–3584.
[49] W.H. Lim, N.A. Mat Isa, Two-layer particle swarm optimization with intelligent division of labor, Eng. Appl. Artif. Intell. 26 (2013) 2327–2348.
[50] W.H. Lim, N.A. Mat Isa, Teaching and peer-learning particle swarm optimization, Appl. Soft Comput. 18 (2014) 39–58.
[51] W.H. Lim, N.A. Mat Isa, Particle swarm optimization with increasing topology connectivity, Eng. Appl. Artif. Intell. 27 (2014) 80–102.
[52] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Computat. 8 (2004) 204–210.
[53] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Computat. 10 (2006) 281–295.
[54] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surv. 45 (2013) 1–33.
[55] C.-L. Huang, W.-C. Huang, H.-Y. Chang, Y.-C. Yeh, C.-Y. Tsai, Hybridization strategies for continuous ant colony optimization and particle swarm optimization applied to data clustering, Appl. Soft Comput. 13 (2013) 3864–3872.
[56] S. Jayaprakasam, S.K.A. Rahim, C.Y. Leow, PSOGSA-explore: a new hybrid metaheuristic approach for beampattern optimization in collaborative beamforming, Appl. Soft Comput. 30 (2015) 229–237.
[57] Y. Liu, B. Niu, Y. Luo, Hybrid learning particle swarm optimizer with genetic disturbance, Neurocomputing 151 (Part 3) (2015) 1237–1247.
[58] M. Mahi, Ö.K. Baykan, H. Kodaz, A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem, Appl. Soft Comput. 30 (2015) 484–490.
[59] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (2011) 4135–4151.
[60] G. Wu, D. Qiu, Y. Yu, W. Pedrycz, M. Ma, H. Li, Superior solution guided particle swarm optimization combined with local search techniques, Expert Syst. Appl. 41 (2014) 7536–7548.
[61] S. Yuhui, R. Eberhart, A modified particle swarm optimizer, in: in: The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, 1998, pp. 69–73.
[62] R.S. Sutton, D. Precup, S. Singh, Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning, Artif. Intell. 112 (1999) 181–211.
[63] M. McPartland, M. Gallagher, Reinforcement learning in first person shooter games, IEEE Trans. Computat. Intell. AI Games 3 (2011) 43–56.
[64] R. Sharma, M.T.J. Spaan, Bayesian-game-based fuzzy reinforcement learning control for decentralized POMDPs, IEEE Trans. Computat. Intell. AI Games 4 (2012) 309–328.
[65] C.C.H. Watkins, P. Dayan, Q-learning, Mach. Learning 8 (1992) 279–292.
[66] T. Huang, A.S. Mohan, Micro-particle swarm optimizer for solving high dimensional optimization problems (uPSO for high dimensional optimization problems), Appl. Math. Computat. 181 (2006) 1148–1154.
[67] A. Ratnaweera, S. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Computat. 8 (2004) 240–255.
[68] C. Li, S. Yang, T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 42 (2012) 627–646.
[69] K. Tang, Z. Li, L. Luo, B. Liu, Multi-strategy adaptive particle swarm optimization for numerical optimization, Eng. Appl. Artif. Intell. 37 (2015) 9–19.
[70] Q.-K. Pan, M. Fatih Tasgetiren, Y.-C. Liang, A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem, Comp. Operat. Res. 35 (2008) 2807–2839.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006

[71] C.-H. Wang, T.-W. Lin, Improved particle swarm optimization to minimize periodic preventive maintenance cost for series-parallel systems, Expert Syst. Appl. 38 (2011) 8963–8969.
[72] Y. Lee, J.J. Filliben, R.J. Micheals, P. Jonathon Phillips, Sensitivity analysis for biometric systems: A methodology based on orthogonal experiment designs, Comp. Vision Image Understand. 117 (2013) 532–550.
[73] M. Črepinšek, S.-H. Liu, M. Mernik, Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them, Appl. Soft Comput. 19 (2014) 161–170.
[74] B. Efron, Bootstrap methods: another look at the Jackknife, Ann. Stat. 7 (1979) 1–26.
[75] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical Report, Nanyang Technological University, Singapore, May 2005 and KanGAL Report 2005005, IIT Kanpur, India, 2005.
[76] Z.-H. Zhan, J. Zhang, Y. Li, Y.-h. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Computat. 15 (2010) 832–847.
[77] W.N. Chen, J. Zhang, Y. Lin, N. Chen, Z.H. Zhan, H.S.H. Chung, Y. Li, Y.H. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Computat. 17 (2013) 241–258.
[78] G. Wu, Across neighborhood search for numerical optimization, Inform. Sci. 329 (2016) 597–618.
[79] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[80] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, CRC Press, 2003.
[81] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (1990) 223–229.
[82] S. Mirjalili, S. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. (2015) 1–19.
[83] A.H. Gandomi, Interior search algorithm (ISA): a novel approach for global optimization, ISA Trans. 53 (2014) 1168–1183.
[84] A. Gandomi, X.-S. Yang, A. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comp. 29 (2013) 17–35.
[85] A. Baykasoğlu, F.B. Ozsoydan, Adaptive firefly algorithm with chaos for mechanical design optimization problems, Appl. Soft Comput. 36 (2015) 152–164.
[86] F. Glover, Future paths for integer programming and links to artificial intelligence, Comp. Oper. Res. (1986). Q5
[87] C.C. Ski, B.L. Golden, Optimization by simulated annealing: a preliminary computational study for the TSP, in: Proceedings of the 15th Conference on Winter Simulation – Volume 2, Arlington, Virginia, USA, IEEE Press, 1983, pp. 523–535.
[88] R. Battiti, M. Brunato, F. Mascia, Reactive Search and Intelligent Optimization, Springer Publishing Company, Incorporated, 2008.
[89] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi, The Bees algorithm. Technical note, Manufacturing Engineering Centre, Cardiff University, UK, 2005, pp. 1–57.
[90] W. Geem, Zong, J. Kim, G. Hoon, V. Loganathan, A new heuristic optimization algorithm: Harmony search, Simulation 76 (2001) 60–68.

Please cite this article in press as: H. Samma, et al., A new Reinforcement Learning-based Memetic Particle Swarm Optimizer, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.01.006