IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 17, 2020, accepted December 29, 2020, date of publication February 5, 2021, date of current version March 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3057515

Multi-Swarm Optimization for Extracting Multiple-Choice Tests From Question Banks

TRAM NGUYEN[1,3], LOAN T. T. NGUYEN[2,6], TOAN BUI[4], HO DAC LOC[4], WITOLD PEDRYCZ[5], (Life Fellow, IEEE), VACLAV SNASEL[3], (Senior Member, IEEE), AND BAY VO[4]

[1]Faculty of Information Technology, Nong Lam University, Ho Chi Minh City 700000, Vietnam
[2]School of Computer Science and Engineering, International University, Ho Chi Minh City 700000, Vietnam
[3]Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VŠB—Technical University of Ostrava, 70800 Ostrava, Czech Republic
[4]Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City 700000, Vietnam
[5]Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6R 2V4, Canada
[6]Vietnam National University, Ho Chi Minh City 700000, Vietnam

Corresponding author: Bay Vo (vd.bay@hutech.edu.vn)

This work was supported by The Youth Incubator for Science and Technology Programe, managed by Youth Development Science and Technology Center - Ho Chi Minh Communist Youth Union and Department of Science and Technology of Ho Chi Minh City, the contract number is "01/HĐ-KHCN-VU".

ABSTRACT In this study, a novel method for generating multiple-choice tests is presented, which extracts the required number of tests of the same levels of difficulty in a single attempt and approximates the difficulty level requirement given by users. We propose an approach using parallelism and Pareto optimization for multi-swarm migration in a particle swarm optimization (PSO) algorithm. Multi-PSO is proposed for shortening the computing time. The proposed migration of PSOs increases the diversity of tests and controls the overlap of extracted tests. The experimental results show that the proposed method can generate many tests from question banks satisfying predefined levels of difficulty. Additionally, the developed method is shown to be effective in terms of many criteria when compared with other methods such as manually extracted tests, a simulated annealing algorithm (SA), random methods and PSO-based approaches in terms of the number of successful solutions, accuracy, standard deviation, search speed, and the number of questions overlapping between the exam questions, as well as for changing the search space, changing the number of individuals, changing the number of swarms, and changing the difficulty requirements.

INDEX TERMS Multiple-choice tests, multi-swarm optimization, multi-objective optimization, parallelism.

I. INTRODUCTION

Education is an essential element for the betterment and progress of a country [38]. Today an important challenge that higher education faces is reaching a stage to enable universities to offer having more efficient, effective and accurate educational processes. An important aspect of education is how to evaluate the learners' progress. There are many methods such as oral tests or writing tests to evaluate students' knowledge and understanding about subjects. Due to the scalability and ease of human resources, writing tests are used more widely for end-of-term evaluations, where a large number of students must be considered. Writing tests can be either descriptive, in which students have to fully write their answers, or multiple-choice tests, in which students pick one or more choices for each question. Although descriptive exams are easier to create at first, they then need a great deal of time and effort from human graders. Multiple choice tests, on the other hand, are harder to create at first as they require a large number of questions for security reasons, as noted in Ting *et al.* [8]. However, the grading process can be extremely fast, automated by computers, and free of the bias that can found with human graders. Recently, many researchers have thus invested efforts to make computers automate the process of creating multiple-choice tests using available question banks [4], [6], [10], [11]. The results have been shown to be promising and thus make multiple-choice tests more feasible for examinations. Although these works have advantages, they still exhibit some drawbacks. First, using a question bank consisting of a large number of items or generating a test that requires many questions is computationally expensive, which leads to poor performance.

The associate editor coordinating the review of this manuscript and approving it for publication was Christian Pilato.

Second, the questions selected randomly may not be evenly spread across the question bank, leading to a lack of diversity. Third, if the question bank consists of multiple-choice items based on objective difficulty levels, then manually choosing items to meet any requirement will take a long time, while the tests produced by a random generation process may not always meet the related requirements. The issue here is generating tests with a single objective, where the level of difficulty is the objective. In this regard, Bui *et al.* [18] proposed the use of Particle Swarm Optimization (PSO) to generate tests by the approximating difficulties to the levels required by users. The tests are generated from question banks that consist of various questions with different difficulties. The difficulty value of each question is judged and adapted based on users via previous real-life exams. The results of their experiments show that PSO gives the best performance with respect to most of the criteria. However, their work only focused on solving a single-swarm single-objective of extracting tests based on the user-defined difficulty level. In practice, when it comes to final exams educational institutions need each student to take a different exam at the same time, to avoid cheating, but all the exams should have the same difficulty, for fairness. This is one of the challenges for generating multiple-choice tests. Furthermore, extracting multiple-choice tests is categorized as a multi-constraint optimization problem based on constraint satisfaction which is NP-hard in [9]. There are many studies which have attempted to solve this problem. However, most of the proposed methods only focus on solving this problem for particular institutes or schools, and deploying these approaches for others is a significant challenge.

For the reasons outlined above, the current paper aims at coming up with solutions to the issue of extracting multiple-choice tests with similar difficulty for multiple students at the same time. With traditional methods such as random search extraction, which most software uses, it is impossible to find the solution to this problem, and the exhaustive search method used requires a very long processing time. Therefore, based on the results of previous research [21], [22], [24], [26] [23], [28], [30], [32], this paper proposed parallel models of heuristic optimization techniques to optimization problems such as PSO to solve the problem of extracting multiple-choice tests from the question banks. Each thread is an algorithm for extracting a test (a swarm), so to extract multiple tests it requires multiple threads to run in parallel. Each swarm now corresponds to a thread and the information exchange happens by chance between swarms in order to improve the convergence and diversity of solutions. Based on [20], we propose a method of using migration theory to allow individual migration between swarms to help the weak ones to increase their abilities to find better solutions, using the method in Lewis [25]. In addition, his combined with the evaluation of objective difficulty can create tests not only of good quality but also of abundant quantity, making them more useful in practice.

In this paper, we propose a method to extract k number of different tests from a question bank in one run. This is an NP-hard combinatorial optimization problem that chooses k tests with the same level of difficulty and guarantees fewer duplicate questions than or an equal number of duplicate questions to an allowed threshold (as described in section III). The main contributions of this paper are outlined as follows:

i We propose a method to deal with the problem of extracting k number of tests simultaneously using multi-PSO.
ii We combine a parallel version of multi-swarm optimization and the theory of migration of dualistic economy to reduce the computing time, increase diversity of tests and control the overlap of extracted tests.
iii We experiment with the proposed extraction method by adjusting the number of individuals and number of swarms. Then we report the results with respect to several essential criteria, including time, stability, and the standard deviation.

The rest of this paper is organized as follows. Some related works are discussed in Section II, while the proposed method is described in Section III. Section IV analyzes the experimental results of this study. Finally, Section V provides some conclusions and suggests some potential directions for future research.

II. LITERATURE REVIEW

A current trend is to use simplified metaheuristic algorithms to deal with complex optimization problems. Some of the most well-known algorithms are the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), as reported in [27], [29], [30], [33].

PSO [1], [2] is an optimization algorithm inspired by the movement of organisms in a bird flock [1] or fish school [3], and each member of the flock or school is called as a particle. PSO is a meta-heuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. All particles will fly in a real-value D dimension search space in an attempt to uncover ever-better solutions to the problem of interest.

Every particle has two attributes: current position X and velocity V. All particles also share two values: the best location in the search space that it has found so far (P_{best}), and the best location found to date by all the particles in the population (G_{best}). At each iteration of the algorithm, particles will update their position and velocity. The magnitude and direction of their velocity is impacted by their velocity in the previous iteration of the algorithm and the location of a particle relative to the location of its P_{best}, and the G_{best}. Thus, PSO is similar to GA, although while GA [37] uses the crossover and mutate functions, PSO uses a function that is the current particles use the experience of previous particles and the social influence of their peer groups.

The following algorithm shows the detail of PSO:
The velocity update function

$$V^{t+1} = V^t + c_1 r_1 \left(P^t_{best} - X^t\right) + c_2 r_2 \left(G^t_{best} - X^t\right)$$

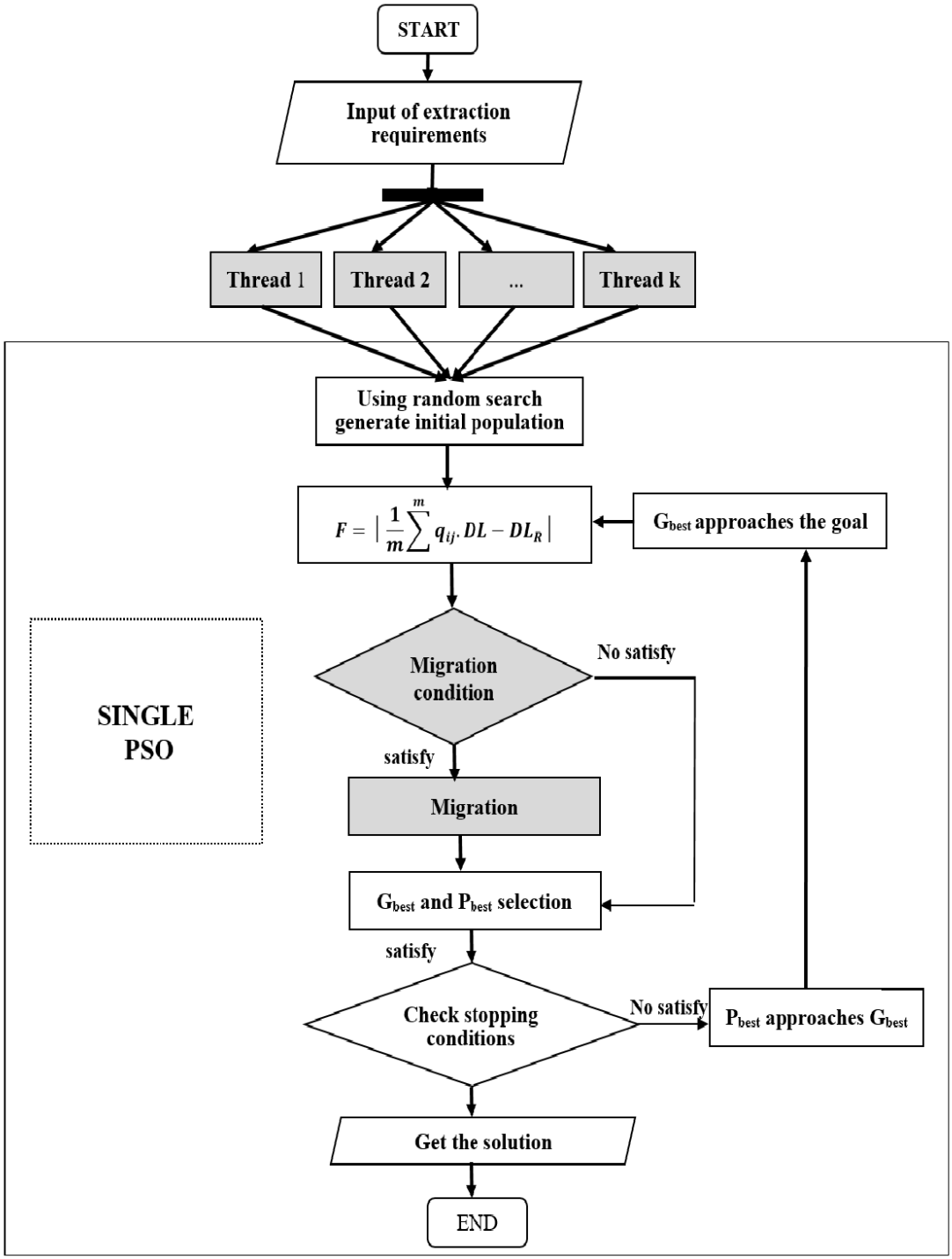

FIGURE 1. The complete flowchart that applies the parallelized migration method to the PSO algorithm.

The position update function

$$X^{t+1} = X^t + V^{t+1}$$

where r_{1d}, r_{2d} is a function that returns a random number in the range (0,1) and c_1, c_2 are constant weights.

Many researchers have proposed numerous methods to extract multiple-choice tests from a question bank, such as randomization [4]–[6] and shuffling algorithm [10], [11]. However, one of the challenges for generating multiple-choice tests is the difficulty of the candidate tests. The tests

TABLE 1. Question bank.

CQ	01	02	03	04	05	06	07	08	09	10
DL	0.3	0.2	0.8	0.7	0.4	0.6	0.5	0.8	0.2	0.3

for all students should have the same difficulty for fairness. However, it can be seen that generating many tests with the same level of difficulty is an extremely hard task, even in the case of manually choosing questions from a question bank, and so the success rate of generating multiple-choice tests satisfying a given difficulty is low and time-consuming. Therefore, to speed up the process and improve the quality of the results, some authors apply local search algorithms such as Tabu Search (TS) [9] and Simulated Annealing (SA) [7] to generate tests with the use of computers and approximate the resulting difficulties to the required ones. Some authors also apply GA [12], [13] to solve the problem of extracting tests, and the experimental results showed that such approaches can reach approximately a 100% success rate, and GA is also faster than SA for this task. PSO can generate multiple tests by optimizing a fitness function which is defined based on multi-criteria constraints [14]–[19]. However, most of the research for single-objective single-solution or multi-objective single-solution extraction of tests are only based on question banks with the subjective difficulty levels for the questions. These can only extract tests which have overall subjective difficulty levels of the questions equal to the difficulty requirements of the users. In addition, extracting multiple-choice tests is categorized as a multi-constraint optimization problem with regard to constraint satisfaction, which is NP-hard [9]. There are many studies, which have tried to solve this problem, but most of the proposed methods only focus on solving it for particular institutes or schools, and so deploying them elsewhere remains a big challenge.

Therefore, in this work, we propose an approach that uses Multi-Swarm Particle Swarm Optimization (MPSO) for generating k tests with a single-objective. Each swarm, in this case, is a test candidate and it runs on a separate thread. The migration happens randomly, by chance. We also aim to improve the accuracy and diversity of the solutions.

III. PROBLEM FORMULATION

This section is divided into five subsections. They cover the problem statement of extracting tests, the fitness function, the overlap in problem solution, the PSO-based method for the problem of extracting tests, and the improvement by using multi-swarm migration and the parallelism of multi-swarm migration in PSO for the problem of extracting tests.

A. PROBLEM STATEMENT

In our previous work [18] we proposed a PSO-based method for multiple-choice test generation, which is a single-objective single-solution approach. In this paper, we introduce a multi-swarm approach of multiple-choice tests generation by combining of PSO and a parallel version of multi-swarm based on the theory of migration of dualistic economy to accelerate the computing time, increase diversity of tests and control their overlap.

The problem of generating k multiple-choice tests is to produce k tests in one attempt. The levels of difficulty for all produced tests in the same attempt must be equivalent and approximate to the specific difficulty level requirement given by the user (denoted by DL_R). The question banks and each question in a bank come with pre-assigned objective difficulty levels by users, and the objective levels of difficulty have a continuous value domain (0,1]. Additionally, an objective difficulty level is assessed based on feedback collected from test-takers that are defined by (1).

Extracting k tests must not only solve the aforementioned requirements for the levels of difficulty given by users, but also arrange logical constraints that relate to the triangle elements (Y, Z, DL_R) where:

Y is the test information matrix to generate a test

Z denotes a set of constraints, $Z = \{z_1, z_2, \ldots, z_l\}$.

DL_R denotes objective difficulty levels set by users.

B. TARGET FUNCTION

Assume that the multiple-choice question bank has m questions $Q = \{q_1, q_2, \ldots, q_m\}$, in which each question has the attributes of question identifier (CQ), part number (CP), and objective difficulty level (DL). Our objective is to generate a test T_i including n questions $(n \leq m) T_i = \{q_{i1}, q_{i2}, \ldots, q_{in}\}$ $(q_{ij} \in Q)$ which satisfy the constraints from a matrix test and the requirement of difficulty level (DL_R).

An objective difficulty level for each question is quantified based on the feedback collected from test-takers. Because the items are evaluated objectively, an item can be computed by

$$DL_i = \frac{the\ number\ of\ \ correct\ answers_i}{total\ number\ of\ answers_i}$$
$$\text{with } DL_i \in (0, 1] \tag{1}$$

For example, question A is taken from a question bank and 50 students (st) take a test in which 30 students correctly question A, and 20 students incorrectly answer it, thus the objective difficulty level of question A is computed by

$$DL_A = \frac{the\ number\ of\ st\ correct_A}{total\ number\ of\ st_A} = \frac{30}{50} = 0.6$$

Therefore, objective difficulty level of question A is 0.6

Additionally, for those questions that have never been attempted by anyone, the objective default difficulty level is 0.5, which is a subjective difficulty, and this will then update to become an objective difficulty based on feedback collected from test-takers that are defined by (1).

The objective difficulty of a test T_i is the average level of the difficulty requirements of the tests, which is defined as

$$\boldsymbol{DL_{T_i}} = \frac{\sum_{j=1}^{n} \boldsymbol{q_{ij}}.\boldsymbol{DL}}{\boldsymbol{n}} \tag{2}$$

The objective function is to ensure the minimum item information, and this is defined as follows

$$\min(f(q) = \left| \frac{\sum_{j=1}^{n} q_{ij}.DL}{n} - DL_R \right|), q \epsilon Q \quad (3)$$

where $f(q)$ is the objective function; n is the total number of questions; $\boldsymbol{q_{ij}.DL}$ is the difficulty of each question; DL_R is the difficulty level requirement; and Q is the solution space.

The objective function $f(q)$ is used as the fitness function in the algorithm, and the results of the objective function are considered as the fitness of the resulting test.

When we extract k tests, the tests have better fitness values and must satisfy the following constraints:

C_1: Each question in a generated test must be unique (i.e., a question cannot appear more than once in a test).

C_2: In order to make the test more diverse, there exists no case that all questions in a test have the same difficulty value as the required objective difficulty DL_R. For example, if $DL_R = 0.6$, then $\exists q_{ki} \in T_k$: $q_{ki}.DL \neq 0.6$.

C3: Some questions in a question bank must stay in the same groups because their content relates to each other. The generated tests must ensure that all the questions in one group appear together. This means if a question of a specific group appears in a test, the remaining questions in the group must also be presented in the same test (Bui *et al.* [18] and Nguyen *et al.* [19]).

C_4: As users may require the generated tests to have several sections, a generated test must ensure that the required number of questions is drawn out from question banks for each section.

C. THE OVERLAP IN PROBLEM SOLUTIONS (THE DIVERSITY OF PROBLEM SOLUTIONS)

To evaluate the quality of the PSO algorithm when it is applied to the problem of extracting k number of tests, it is expressed through the estimation of an overlap percentage P_T of questions in each test of k tests. A test is acceptable when P_T is less than or equal to the threshold (we set this as below 30% in our experiments). Formula (4) shows the way to calculate P_T.

$$P_T = \frac{\sum_{i=1}^{n} \sum_{j=1, ji}^{n} D_{ij}}{k} \quad (4)$$

where:

P_T: the overlap percentage of k tests. The smaller the value is, the more optimal the tests are.

D_{ij}: the number of overlap questions of test i and test j.

k: the number of tests, and n is the number of questions in each test.

D. PROPOSED PARTICLE SWARM OPTIMIZATION ALGORITHM WITH MODIFIED VELOCITY UPDATING MECHANISM FOR EXTRACTING MULTIPLE-CHOICE TESTS

The PSO algorithm for extracting multiple-choice tests is described as follows:

Creating an initial swarm population is the first step in PSO, in which each particle in a swarm is considered a candidate test; this first population also affects the speed of convergence to optimal solutions. This step randomly picks questions in a question bank. The questions, either standalone or staying in groups (constraint C_3), are drawn out for one section (constraint C_4) until the number of questions required for the section is reached, and then the drawing process is repeated for next sections. When the required number of questions of the candidate test and all the constraints are met, the fitness value of the generated test will be computed according to formula (3).

A modified particle swarm optimization algorithm based on the velocity updating mechanism is derived as follows.

The G_{best} and P_{best} position information is the questions the test contains. All P_{best} slowly move towards G_{best} by using the location information of G_{best}. The movement is the replacement of some questions in the candidate test according to the velocity P_{best}. If the fitness value of a newly found P_{best} of a particle is smaller than the particle's currently best known P_{best} (i.e., the new position is better than the old one) then we assign the newly found position value to P_{best}.

G_{best} moves towards the final optimal solution in random directions. The movement is achieved by replacing its content with some random questions from the question bank. In a similar way to P_{best}, if the new position is no better than the old one, the G_{best} value will not be updated.

The algorithm ends when the fitness value is lower than the fitness threshold ε or the number of movements (iteration loops) surpasses the loop threshold λ. Both of the thresholds are given by users.

The particles are G_{best} and P_{best}, they update the independent velocity (V) and positions (P) in the extracted tests by using modified PSO algorithm based on the velocity updating mechanism as follows.

We propose a modified velocity updating mechanism based on PSO for extracting multiple-choice tests using the following formulas:

- The velocity of P_{best} ($V_{P_{best}}$) approaches G_{best}, and it is given as:

$$\boldsymbol{V_{P_{best}}} = \boldsymbol{\alpha} \times \boldsymbol{m} \quad (5)$$

- The velocity of G_{best}($V_{G_{best}}$) approaches the goal, and it is given as:

$$\boldsymbol{V_{G_{best}}} = \boldsymbol{\beta} \times \boldsymbol{m} \quad (6)$$

where: $\alpha, \beta \in (0, 1)$ and m is the number of questions in the test solutions.

The position of particles is P_{best} and G_{best}, and this is based on their corresponding velocity.

The modified PSO approach to extract a test is described in the form of a pseudocode as follows:

In order to clearly demonstrate how the modified PSO extracts a multiple-choice test from a question bank, we present the following example.

Pseudocode of modified PSO for extracting a test

Step 1. Use a randomizer to generate an initial swarm population in which each particle in a swarm is considered a candidate test;

LOOP

Step 2. Select G_{best} and P_{best};

Step 3. IF G_{best} satisfies the conditions stop THEN exit LOOP.

Step 4. Update the locations of individuals

Step 4.1. The P_{best} individuals' approach G_{best}, with V_{Pbest} (5)

'Step 4.2. G_{best} approaches the goal, with V_{Gbest} (6) G_{best} moves in a random direction to search for the optimal solution;

END LOOP

We have a question bank as shown in Table 1. Using the modified PSO to order the test extraction requirements are four questions, with a difficulty level of 0.3 (DL_R =0.3); coefficients: $\alpha = 0.25$, $\beta = 0.25$. A fitness smaller than $\varepsilon = 0.003$ is taken as the stopping criterion.

- ***Generation 1, initialize a population that includes two individuals***:

Individual 1	**CQ**	05	08	01	04	**Fitness**
	DL	0.4	0.8	0.3	0.7	0.25
Individual 2	**CQ**	02	06	01	03	**Fitness**
	DL	0.2	0.6	0.3	0.8	0.175

- Selection of G_{best} and P_{best}:

Individual 2 G_{best}	**CQ**	02	06	01	03	**Fitness**
	DL	0.2	0.6	0.3	0.8	0.175
Individual 1 P_{best}	**CQ**	05	08	01	04	**Fitness**
	DL	0.4	0.8	0.3	0.7	0.25

Individual G_{best} has the fitness of 0.175, which does not satisfy the stopping condition.

- G_{best} approaches the goal by receiving a random question from the question bank:

Individual 4	**CQ**	02	**04**	01	03	**Fitness**
	DL	0.2	**0.7**	0.3	0.8	0.2

- P_{best} approaches G_{best} by receiving a random question 06 from G_{best}:

Individual 3	**CQ**	05	**06**	01	04	**Fitness**
	DL	0.4	**0.6**	0.3	0.7	0.2

- ***Generation*** 2, selection of G_{best} and P_{best}:

Individual 2 G_{best}	**CQ**	02	06	01	03	**Fitness**
	DL	0.2	0.6	0.3	0.8	0.175
Individual 3 P_{best}	**CQ**	05	06	01	04	**Fitness**
	DL	0.4	0.6	0.3	0.7	0.2

- G_{best} approaches the goal by receiving a random question from question bank:

Individual 5	**CQ**	02	06	01	**10**	**Fitness**
	DL	0.2	0.6	0.3	**0.3**	0.05

- P_{best} approaches G_{best} by receiving a random question 02 from G_{best}:

Individual 6	**CQ**	**02**	06	01	04	**Fitness**
	DL	**0.2**	0.6	0.3	0.7	0.15

- ***Generation*** 3, selection of G_{best} and P_{best}:

Individual 5 G_{best}	**CQ**	02	06	01	10	**Fitness**
	DL	0.2	0.6	0.3	0.3	0.05
Individual 6 P_{best}	**CQ**	02	06	01	04	**Fitness**
	DL	0.2	0.6	0.3	0.7	0.15

- G_{best} approaches the goal by receiving a random question from question bank:

Individual 7	**CQ**	02	**05**	01	10	**Fitness**
	DL	0.2	**0.4**	0.3	0.3	0

- P_{best} approaches G_{best} by receiving a random question 10 from G_{best}:

Individual 8	**CQ**	02	06	01	**10**	**Fitness**
	DL	0.2	0.6	0.3	**0.3**	0.05

- ***Generation*** 3, selection G_{best} and P_{best}:

Individual 7 G_{best}	**CQ**	02	05	01	10	**Fitness**
	DL	0.2	0.4	0.3	0.3	0
Individual 8 P_{best}	**CQ**	02	06	01	10	**Fitness**
	DL	0.2	0.6	0.3	0.3	0.05

The fitness value of G_{best} is 0, and thus satisfies the stopping condition. As such, the test extraction requirement is met by Individual 7.

E. IMPROVING PSO-BASED METHOD BY USING MULTI-SWARM MIGRATION AND PARALLELISM

The problem of extracting tests from question banks often deals with a very large search space. we thus present a novel method for the generation of the multiple-choice tests, which extracts an abundance of tests at the same time with equivalent levels of difficulty and approximates the specific difficulty level requirement given by the user. The proposed approach for extracting tests from a question bank is based on the parallelism of multi-swarm migration PSO to improve the runtime, accuracy and density of the results. This approach is novel and performs better than the traditional approaches, such as manually extracted tests, random methods, simulated annealing and other methods based on the PSO algorithm.

1) MULTI-SWARM MIGRATION IN PSO ALGORITHM

In [25], the Lewis "Two-Sector" migration theory describes the relationship between two traditional rural and urban industrial economies, explaining the movement of surplus labor in the area. Individuals in the traditional rural economy move to the modern urban industrial economy (while ignoring other related aspects of the economy), and whether a sector is strong or weak depends on the fitness value of G_{best} positions of its swarm. Based on the Lewis "Two-Sector" migration theory and Pareto optimization, in this paper we make some adjustments so that the parallel migration can yield more optimal solutions. The direction of migration changes when individuals with the second strong P_{best} (strong individuals) in strong sectors move to weak sectors. The weak sectors' G_{best} may be replaced by the incoming P_{best}, and the fitness value of the weak swarms should rise considerably. Backward migration from the weak swarms to strong swarms also happens alongside forward migration. For each migratory individual, there will be one migrating back to the swarm in exchange. This is to ensure that the number of particles and

the searching capabilities of the swarms do not significantly decrease.

The condition for migration in the swarm is when the fitness value of G_{best} in the current generation is different from that of G_{best} in the previous generation (better fitness G_{best} in the previous generation).

2) MIGRATION MODEL

The migration model is expressed as follows:

$$\theta = min(F_1, F_2, F_3, \ldots F_u) \tag{7}$$

With u being the number of generations and $\theta \in [0, 1)$, the smaller θ is the better the fitness

$$F_i = \frac{\sum_{j=1}^{m} q_{ij}.DL}{m} - DL_R$$
$$\text{where } \mathrm{DL_R} \in (0, 1]. \tag{8}$$

As m stands for the questions in the test, $q_{ij}.DL$ is the difficulty of each question and DL_R is the difficulty level requirement that is defined by the user.

The migration probability is denoted by :

$$\phi = m \times \varphi \tag{9}$$

with n is defined in (8), φ is a scaling factor with the value defined by the user.

The number of migrating individuals is calculated as follows:

$$\lambda = t \times \phi \tag{10}$$

with φ in (9), t is the number of individuals in the swarm.

The migration process chooses the smallest θ (the best fitness) by moving target immigration that has the biggest θ (fitness is the weakest) satisfying (7) (8) (9) (10).

The migration parallel MPSO approach to extract multiple tests is described in a form of pseudocode in the following algorithm:

3) THE PARALLELISM OF MULTI-SWARM MIGRATION IN PSO ALGORITHM

The multi-swarm migration uses multi-threads to perform parallelism. Each thread corresponds to a thread running the PSO algorithm. However, the active process of migration induces a probability of migration, which represents the migration between swarms. The initial process locks the current thread in order to avoid interference from other threads to the current thread. The process selects a weaker swarm than the current one to avoid the complete exchange of swarms, so the variable ''Exchanging'' is used as a flag to avoid this from happening.

In order to improve the quality of the solution, we should select the parameter values of migration between swarms (migration probability γ, migration scale δ), which are important and impact the quality of the solution. In this paper, we select them between [5%, 10%]. There is guaranteed Pareto optimization for the migration problem between the strong swarm and the weakest swarm (weak area) in extracting k tests. The experimental results show that the selection parameters of migration are efficient, such as high-speed convergence, diversity of solutions. Conversely, any selection that serves for moving to widespread migration will lead to suboptimal solutions, or even cannot find a solution. The complete flowchart that applies the parallelized migration method to the PSO algorithm is shown in Figure 1.

The parallelism of multi-swarm migration in the PSO algorithm in Figure 1, is as follows:

The multi-swarm migration uses multi-threads to perform parallelism. Each thread corresponds to a thread running the PSO algorithm. In the first stage the algorithm proceeds to find tests that satisfy all requirements and constraints using

Pseudocode: Migration Parallel MMPSO for extracting tests

For each available thread t do
Step 1. Use a randomizer to generate an initial swarm population in which each particle in a swarm is considered a candidate test;
LOOP
Step 2. Select G_{best} and P_{best};
Step 3. IF G_{best} satisfies the conditions stop THEN exit LOOP.
Step 4. Update the locations of individuals
Step 4.1. The P_{best} individuals approaches G_{best}, with V_{Pbest} (5)
'Step 4.2. Gb_{est} approaches the goal, with V_{Gbest} (6)
G_{best} moves in a random direction to search for the optimal solution;
If the probability for migration γ is met **then**
Execute function **Migration_MMPSO** with t;
Endif
END LOOP
ENDFOR
Function Migration _MPSO: Improving solutions with the migration method
Lock the current thread (i.e., block all modifications from other threads to the current thread) to avoid interference from other threads to the current thread during the migration procedure.
Select λ, which are the set of stronger individuals for migration except for the G_{best};
Lock the other threads:
Choose a thread that has a G_{best} weaker than the one in the current thread;
Unlock the other threads except for the chosen thread;
Set the status of the chosen thread to ''Exchanging'';
Move the λ selected individuals to the chosen thread;
Remove those λ selected individuals;
Select the λ weakest individuals in the chosen thread;
Add those λ weakest individuals to the current thread;
Set the status of chosen thread to ''Available'';
Unlock the current thread and the chosen thread;

TABLE 2. Experimental parameters.

The required level of difficulty (ODR)	**[0.1, 0.9]**
The number of questions required in a test	**100**
The number of questions in each section in the test	**10**
The number of simultaneously generated tests in each run (the number of swarms)	**[100, 200, 400, 600, 800, 1000]**
The number of questions in the bank	**1,000 and 12,000**
The PSO's parameters	**The number of particles in each swarm: [10, 20, 40, 60, 80, 100, 120,140, 160, 180, 200]** **Random value r_1, r_2 are in [0,1];** **The percentage of P_{best} individuals which receive position information from G_{best} (C_1): 5%;** **The percentage of G_{best} which move to the final goals (C_2): 5%;** **The percentage of migration δ : 10%;** **The percentage of migration probability γ : 5%** **The stop condition: either when the tolerance fitness < 0.0001 or when the number of movement loops > 1000;**
The SA's parameters	**Initial temperature: 100;** **Cooling rate: 0.9;** **Termination temperature: 0.0001;** **Number of iterations: 1000;**

multiple threads. Each thread corresponds to each swarm that runs separately. The second stage is improving and diversifying tests. This stage happens when there is a change in the value of G_{best} of each swarm (for each thread) in the first stage. In this second stage, migration happens between swarms to exchange information between running threads to improve the convergence and diversity of solutions.

The condition for migration in the swarm is when the fitness values of G_{best} in the current generation are different from those of G_{best} in the previous generation (better fitness G_{best} in the previous generation).

IV. EXPERIMENTAL STUDIES

To ensure fairness between the tests, each test should be extracted from a question bank with similar difficulty and each student should have a different test. This problem is important when extracting multiple tests with similar difficulty for multiple students at the same time. We compare the efficiency of algorithms such as PSO following sequential, parallel and parallel migration; Simulate Annealing Algorithm [7]; and Random Algorithm [4], using the same criteria for the experiments. Based on the average statistical method after 10 runs, we examine the number of successful solutions, accuracy, standard deviation, search speed, and number of questions overlapping between the exam questions, as well as changing the search space (large and small question banks), changing the number of individuals, changing the number of groups (number of questions), and changing the difficulty requirements of the test.

The experimental data includes two question banks. One has 1,000 different questions (the small question bank) and the other 12,000 different questions (the large question bank). The small question bank consists of multiple sections and each section has more than 150 questions with different difficulty levels (Figure 2). The large question bank includes 12,000 different questions in which each part has 1,000 questions with different difficulty levels (Figure 3). The experimental parameters of PSO [36] are presented in Table 2. The results are shown in Tables 3 to 8.

Regarding the results, each run extracts them by changing a large number of tests simultaneously and each test has a fitness value. Each run also requires a number of iteration loops to successfully extract the required tests. The average runtime for extracting tests is the average of the runtimes of all 10 experimental runs. The average fitness is the average of all fitness values of a large number of tests generated from 10 runs. The average duplicate indicates the average number of duplicate questions among a large number of tests generated from all 10 runs. The average duplicate is also used to indicate the diversity of tests. The lower the value, the more diverse the tests.

A. EXPERIMENTAL ENVIRONMENT

The algorithms are implemented in C#, (Microsoft Visual Studio 2013), using Windows 8.1 Operating System, and running on a computer with a 2.5GHz of CPU and 4 GB RAM.

TABLE 3. Experimental results for changing numbers of individuals in the large question bank.

Algorithms	The number of individuals	The number of swarms	Difficult level	Number of runs	Successful solutions	Average runtime for extracting tests (seconds)	Average fitness	Average duplicate (%)	Standard deviation
Sequential PSO	**10**	100	*0.5*	10	1000	9.27	0.0000475833	0.90	0.000028287
	20	100	*0.5*	10	1000	13.21	0.0000493549	0.90	0.0000274907
	40	100	*0.5*	10	1000	19.68	0.000048586	0.90	0.0000286029
	60	100	*0.5*	10	1000	26.88	0.0000466945	0.90	0.0000286299
	80	100	*0.5*	10	1000	32.14	0.0000486801	0.90	0.0000291263
	100	100	*0.5*	10	1000	37.19	0.0000464449	0.90	0.0000286816
	120	100	*0.5*	10	1000	45.12	0.0000463716	0.90	0.000029507
	140	100	*0.5*	10	1000	51.92	0.0000456173	0.90	0.000029046
	160	100	*0.5*	10	1000	55.62	0.0000450677	0.90	0.0000280032
	180	100	*0.5*	10	1000	63.95	0.0000457174	0.90	0.0000283821
	200	100	*0.5*	10	1000	68.67	0.0000439222	0.90	0.0000280759
Parallel PSO	**10**	100	*0.5*	10	1000	4.18	0.0000477304	0.90	0.0000289519
	20	100	*0.5*	10	1000	7.84	0.000049483	0.90	0.0000292806
	40	100	*0.5*	10	1000	14.92	0.0000477106	0.90	0.0000290143
	60	100	*0.5*	10	1000	21.31	0.0000475484	0.90	0.0000286469
	80	100	*0.5*	10	1000	28.66	0.0000466015	0.90	0.0000286927
	100	100	*0.5*	10	1000	37.01	0.0000462559	0.90	0.0000285836
	120	100	*0.5*	10	1000	43.95	0.00004626	0.90	0.0000285891
	140	100	*0.5*	10	1000	50.86	0.000045679	0.90	0.0000282073
	160	100	*0.5*	10	1000	56.8	0.0000449725	0.91	0.0000286526
	180	100	*0.5*	10	1000	63.79	0.000044487	0.89	0.0000295167
	200	100	*0.5*	10	1000	70.81	0.0000450223	0.90	0.0000279527
Parallel Migration PSO	**10**	100	*0.5*	10	1000	4.01	0.0000506994	0.90	0.0000287676
	20	100	*0.5*	10	1000	7.21	0.0000499046	0.89	0.0000283824
	40	100	*0.5*	10	1000	13.77	0.0000484824	0.90	0.0000289764
	60	100	*0.5*	10	1000	18.69	0.0000484881	0.89	0.0000291892
	80	100	*0.5*	10	1000	24.65	0.0000469823	0.89	0.0000288724
	100	100	*0.5*	10	1000	30.21	0.000047517	0.90	0.000028687
	120	100	*0.5*	10	1000	41.1	0.0000466806	0.91	0.0000283357
	140	100	*0.5*	10	1000	47.74	0.0000465141	0.91	0.0000292992
	160	100	*0.5*	10	1000	58.62	0.0000456067	0.90	0.0000280124
	180	100	*0.5*	10	1000	66.64	0.0000444451	0.90	0.0000293941
	200	100	*0.5*	10	1000	70.24	0.0000433255	0.90	0.0000284653
Random Algorithm [4]	**10**	100	*0.5*	10	32	32.72	0.0029871794	0.90	0.0027284722
	20	100	*0.5*	10	52	54.01	0.0016628545	0.90	0.001503191
	40	100	*0.5*	10	112	84.39	0.0007790784	0.90	0.0008087782
	60	100	*0.5*	10	162	129.03	0.0005769422	0.91	0.000568634
	80	100	*0.5*	10	204	166.46	0.000400958	0.90	0.0003859342
	100	100	*0.5*	10	279	188.56	0.0003168756	0.89	0.0003194892
	120	100	*0.5*	10	329	206.53	0.0002596932	0.89	0.0002537287

TABLE 3. *(Continued.)* **Experimental results for changing numbers of individuals in the large question bank.**

	140	100	*0.5*	10	353	286.91	0.0002256974	0.90	0.0002186149
	160	100	*0.5*	10	383	322.47	0.0002032915	0.90	0.0002072332
	180	100	*0.5*	10	421	356	0.0001922599	0.90	0.0001893135
	200	100	*0.5*	10	460	397.64	0.0001602609	0.90	0.000164197
Simulated Annealing Algorithm (SA) [7]	**10**	100	*0.5*	10	187	63.11	0.0008725566	0.91	0.0009278136
	20	100	*0.5*	10	174	63.29	0.0009224878	0.90	0.0009694452
	40	100	*0.5*	10	173	62.65	0.0009293273	0.90	0.0009579109
	60	100	*0.5*	10	169	64.58	0.0008841845	0.90	0.0008827386
	80	100	*0.5*	10	187	63.31	0.000920168	0.90	0.0009930869
	100	100	*0.5*	10	194	62.17	0.0008983694	0.89	0.0009152836
	120	100	*0.5*	10	190	62.69	0.0008599878	0.90	0.0008618551
	140	100	*0.5*	10	173	62.94	0.0008785079	0.90	0.0009322298
	160	100	*0.5*	10	176	62.91	0.0008929886	0.90	0.0009304746
	180	100	*0.5*	10	191	62.43	0.0008562141	0.90	0.000888402
	200	100	*0.5*	10	192	62.57	0.0009092435	0.90	0.0009533097

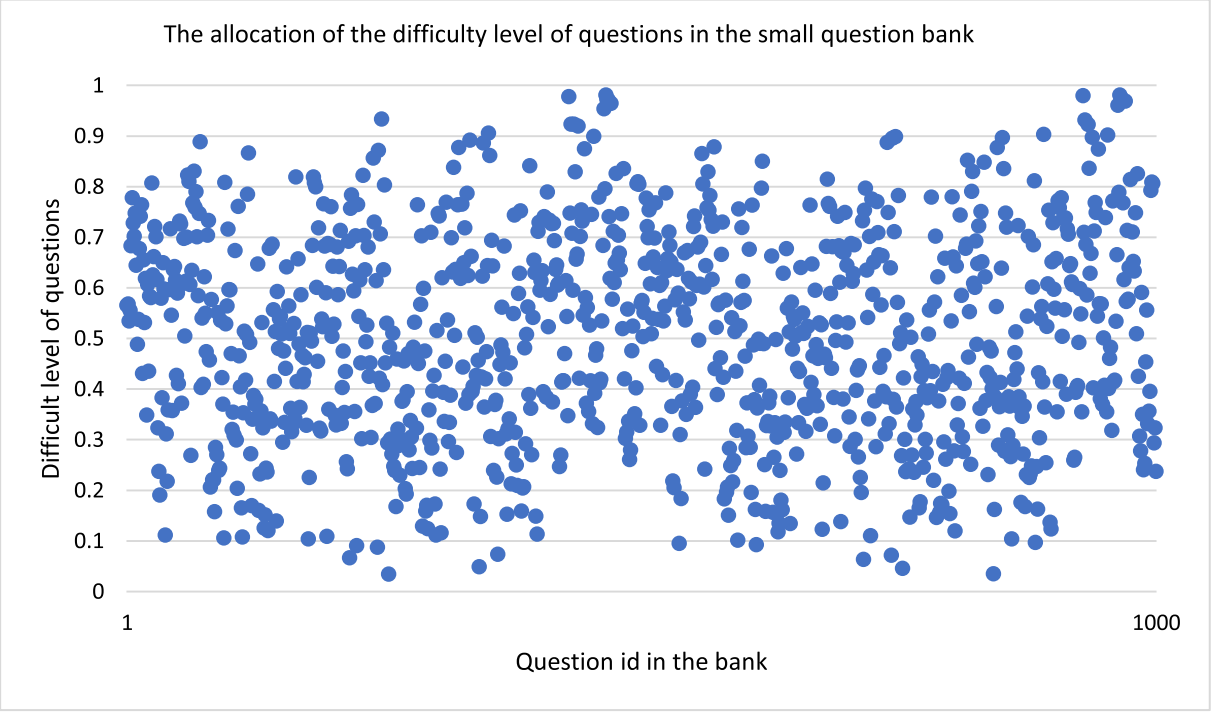

FIGURE 2. The allocation of the difficulty levels of questions in the small question bank.

Our experiments focus on implementing the formula (3), which derives the average levels of difficulty requirements of the tests.

Parameter values for all experiments are taken from Table 2.

B. CHANGING NUMBERS OF INDIVIDUALS IN SWARMS

In this section, we present evaluations for the algorithms regarding the stability when the level of difficulty is 0.5. In this, we prove that PSO methods find good solutions in huge search spaces.

These experiments aim to find the number of individuals of each swarm that will help the algorithm performs its best.

1) LARGE QUESTION BANK (LARGE SPACE SEARCH)

The experimental results of the algorithms after 10 runs show that when we increased the number of individuals in the swarm of both methods there was a decline in the standard deviation at the same rate. Furthermore, the runtime of both

TABLE 4. Experimental results of changing numbers of individuals in the small question bank.

Algorithms	The number of individuals	The number of swarms	Difficult level	Number of runs	Successful solutions	Average runtime for extracting tests (second)	Average fitness	Average duplicate (%)	Standard deviation
Sequential PSO	**10**	100	*0.5*	10	1000	5.46	0.0000485788	16.36	0.0000294872
	20	100	*0.5*	10	1000	7.63	0.0000491318	16.35	0.0000284884
	40	100	*0.5*	10	1000	10.16	0.000048263	16.37	0.0000285867
	60	100	*0.5*	10	1000	11.31	0.0000489896	16.35	0.0000288344
	80	100	*0.5*	10	1000	13.59	0.0000483362	16.36	0.0000285393
	100	100	*0.5*	10	1000	10.43	0.0000467882	16.36	0.0000286272
	120	100	*0.5*	10	1000	15.21	0.0000448535	16.35	0.0000287417
	140	100	*0.5*	10	1000	15.74	0.0000475482	16.35	0.0000287489
	160	100	*0.5*	10	1000	17.19	0.0000449274	16.40	0.0000284039
	180	100	*0.5*	10	1000	18.1	0.0000465619	16.35	0.0000285808
	200	100	*0.5*	10	1000	19.4	0.0000461644	16.39	0.0000286931
Parallel PSO	**10**	100	*0.5*	10	1000	1.45	0.0000490743	16.38	0.0000284969
	20	100	*0.5*	10	1000	2.09	0.0000485699	16.37	0.0000287301
	40	100	*0.5*	10	1000	2.94	0.0000476597	16.37	0.0000295002
	60	100	*0.5*	10	1000	3.52	0.0000479225	16.36	0.0000285087
	80	100	*0.5*	10	1000	4.26	0.0000466083	16.36	0.0000291629
	100	100	*0.5*	10	1000	4.9	0.0000474935	16.36	0.0000293683
	120	100	*0.5*	10	1000	5.35	0.0000476945	16.36	0.0000286132
	140	100	*0.5*	10	1000	5.76	0.0000454825	16.37	0.0000283482
	160	100	*0.5*	10	1000	6.12	0.0000489147	16.36	0.0000290475
	180	100	*0.5*	10	1000	6.43	0.0000462391	16.39	0.0000287697
	200	100	*0.5*	10	1000	7.13	0.0000463028	16.36	0.0000289394
Parallel Migration PSO	**10**	100	*0.5*	10	1000	1.38	0.0000502326	16.36	0.0000282002
	20	100	*0.5*	10	1000	1.81	0.0000486829	16.34	0.0000285621
	40	100	*0.5*	10	1000	2.71	0.0000485772	16.38	0.0000282944
	60	100	*0.5*	10	1000	3.58	0.0000474141	16.36	0.0000288159
	80	100	*0.5*	10	1000	4.56	0.0000464707	16.38	0.0000286011
	100	100	*0.5*	10	1000	4.84	0.0000469956	16.38	0.0000288997
	120	100	*0.5*	10	1000	5.5	0.0000473538	16.37	0.0000293038
	140	100	*0.5*	10	1000	6.6	0.00004609	16.37	0.0000291581
	160	100	*0.5*	10	1000	7.12	0.000047141	16.36	0.0000287308
	180	100	*0.5*	10	1000	7.97	0.0000461727	16.38	0.0000282494
	200	100	*0.5*	10	1000	7.88	0.0000460534	16.38	0.0000287818
Random Algorithm [4]	**10**	100	*0.5*	10	13	2.46	0.0057252527	16.29	0.0047719024
	20	100	*0.5*	10	24	3.88	0.0032980439	16.34	0.0029639127
	40	100	*0.5*	10	47	7.16	0.0017181989	16.38	0.0015965904
	60	100	*0.5*	10	89	11.12	0.0011470363	16.35	0.0010965322
	80	100	*0.5*	10	104	13.71	0.0008501147	16.34	0.0008525693

TABLE 4. *(Continued.)* **Experimental results of changing numbers of individuals in the small question bank.**

	100	100	*0.5*	10	140	18.24	0.0006748943	16.37	0.0007067955
	120	100	*0.5*	10	147	20.76	0.0005880672	16.36	0.0005641431
	140	100	*0.5*	10	173	22.4	0.0004892003	16.36	0.0004926219
	160	100	*0.5*	10	215	27.34	0.0004322328	16.37	0.0004486355
	180	100	*0.5*	10	222	29.17	0.0003919978	16.39	0.0003715946
	200	100	*0.5*	10	250	32.63	0.0003593847	16.34	0.0003587486
Simulated Annealing Algorithm [7]	**10**	100	0.5	10	115	5.12	0.0015175529	16.37	0.0015152884
	20	100	0.5	10	99	5.11	0.001480097	16.36	0.0015100509
	40	100	0.5	10	108	5.32	0.0015752111	16.36	0.0014991383
	60	100	0.5	10	89	5.25	0.0015581609	16.37	0.0014904079
	80	100	0.5	10	93	5.37	0.0015432605	16.39	0.0015467513
	100	100	0.5	10	90	5.19	0.0016645228	16.33	0.0017063568
	120	100	0.5	10	92	5.11	0.0016152642	16.33	0.0015446802
	140	100	0.5	10	85	5.22	0.0015963398	16.34	0.0015733574
	160	100	0.5	10	100	5.15	0.0015382702	16.36	0.0015290549
	180	100	0.5	10	86	5.09	0.0015875156	16.36	0.0015790137
	200	100	0.5	10	97	5.12	0.0015432396	16.35	0.0015110477

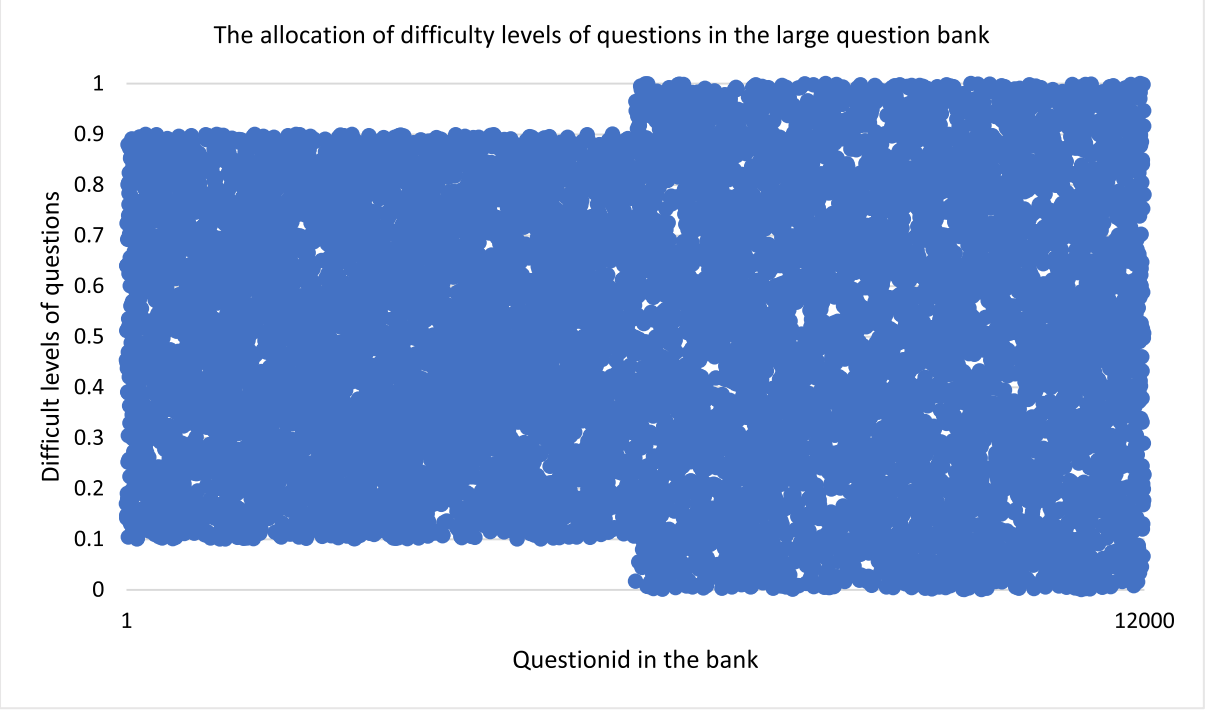

FIGURE 3. The allocation of difficulty levels of questions in the large question bank.

methods increased when the number of individuals in the swarms increased in Table 3. Simulated Annealing has better results than random on some criteria, such as number of successful solutions, runtime and fitness value. However, PSO methods always give better results than SA for all criteria. In the PSO methods, serial PSO takes more time than parallel PSO and migrating parallel PSO. Migrating parallel PSO has better performance than parallel PSO in terms of execution

TABLE 5. Experimental results of changing numbers of swarms in the large question bank.

Algorithms	The number of *swarms*	The number of individuals	Difficult level	Number of runs	Successful solutions	Average runtime for extracting tests (second)	Average fitness	Average duplicate (%)	Standard deviation
Sequential PSO	**100**	50	0.5	10	1000	23.26	0.0000472395	0.90	0.0000286476
	200	50	0.5	10	2000	44.8	0.0000477435	0.90	0.0000288388
	400	50	0.5	10	4000	86.11	0.0000478514	0.90	0.0000287717
	600	50	0.5	10	6000	128.7	0.0000480706	0.90	0.0000288401
	800	50	0.5	10	8000	165.82	0.0000479367	0.90	0.0000288821
	1000	50	0.5	10	10000	202	0.0000483328	0.90	0.0000289065
Parallel PSO	**100**	50	0.5	10	1000	16.83	0.0000465153	**0.92**	0.0000280296
	200	50	0.5	10	2000	34.67	0.0000468262	0.90	0.0000289542
	400	50	0.5	10	4000	78.15	0.0000483981	0.90	0.0000286759
	600	50	0.5	10	6000	100.68	0.0000484697	0.90	0.0000287467
	800	50	0.5	10	8000	121.86	0.0000479001	0.90	0.0000290576
	1000	50	0.5	10	10000	155.91	0.0000480649	0.90	0.0000288733
Parallel Migration PSO	**100**	50	0.5	10	1000	19.28	0.0000493758	0.90	0.0000292478
	200	50	0.5	10	2000	44.51	0.0000474914	0.90	0.0000289322
	400	50	0.5	10	4000	71.73	0.0000476021	0.90	0.0000288632
	600	50	0.5	10	6000	91.96	0.0000472607	0.90	0.0000287841
	800	50	0.5	10	8000	125.78	0.0000482725	0.90	0.0000287967
	1000	50	0.5	10	10000	153.05	0.0000481303	0.90	0.0000289165
Random Algorithm [4]	**100**	50	0.5	10	151	107.67	0.000641586	0.90	0.0006404184
	200	50	0.5	10	263	203.58	0.0006555891	0.90	0.000639235
	400	50	0.5	10	552	304.01	0.0006275829	0.90	0.0006296122
	600	50	0.5	10	863	541.78	0.0006404427	0.90	0.0006338492
	800	50	0.5	10	1146	832.48	0.0006461646	0.90	0.0006405529
	1000	50	0.5	10	1401	1043.43	0.0006443684	0.90	0.0006352166
Simulated Annealing Algorithm [7]	**100**	50	0.5	10	178	64.54	0.0009070393	0.90	0.0009486412
	200	50	0.5	10	373	126.84	0.0008935623	0.90	0.0009291475
	400	50	0.5	10	711	249.79	0.0008944092	0.90	0.0008925994
	600	50	0.5	10	1159	382.05	0.0008742549	0.90	0.0009136319
	800	50	0.5	10	1482	491.04	0.000879932	0.90	0.0009090217
	1000	50	0.5	10	1870	619.11	0.0008838192	0.90	0.0009211429

time, the diversity of solutions, and standard deviation over generations.

2) SMALL QUESTION BANK (SMALL SPACE SEARCH)

With the small question bank the search time is faster, but most of the criteria are not as good as with the large one. However, they still meet all the conditions of the problem. The results in Table 4 show that the success rate for Random and SA is low. In contrast, the PSO methods' success rate still guarantees 100%. With a small search space, we show that parallel migration PSO is more efficient than parallel PSO and sequential PSO in terms of execution time, fitness average, and standard deviation.

In short, given the small or large search space of the question bank, PSO methods always give better results than SA and Random in Tables 3 and 4. This experiment helps decide on the number of individuals for each swarm for extracting tests in the online deployment, which is between 40 and 60.

C. EFFECT OF NUMBER OF SWARMS

In this section, we present evaluations for algorithms regarding the stability when the level of difficulty requirement is

TABLE 6. Experimental results of changing numbers of swarms in the small question bank.

Algorithms	The number of *swarms*	The number of individuals	Difficult level	Number of runs	Successful solutions	Average runtime for extracting tests (second)	Average fitness	Average duplicate (%)	Standard deviation
Sequential PSO	**100**	50	0.5	10	1000	10.37	0.0000472491	16.38	0.0000284972
	200	50	0.5	10	2000	19.35	0.0000478532	16.36	0.0000288602
	400	50	0.5	10	4000	39.9	0.000048561	16.37	0.0000289575
	600	50	0.5	10	6000	59.6	0.0000485378	16.37	0.0000284544
	800	50	0.5	10	8000	79.57	0.0000481051	16.37	0.0000289799
	1000	50	0.5	10	10000	94.12	0.0000476358	16.37	0.00002884
Parallel PSO	**100**	50	0.5	10	1000	2.81	0.0000479555	16.38	0.0000291595
	200	50	0.5	10	2000	5.94	0.0000482158	16.35	0.0000289604
	400	50	0.5	10	4000	11.95	0.0000492851	16.37	0.0000287574
	600	50	0.5	10	6000	17.78	0.0000481608	16.36	0.0000289531
	800	50	0.5	10	8000	22.81	0.0000483982	16.37	0.0000289413
	1000	50	0.5	10	10000	29.5	0.000048291	16.36	0.0000288598
Parallel Migration PSO	**100**	50	0.5	10	1000	2.81	0.0000491176	16.35	0.0000285017
	200	50	0.5	10	2000	5.7	0.000047928	16.38	0.0000292795
	400	50	0.5	10	4000	11.49	0.0000480467	16.36	0.0000289465
	600	50	0.5	10	6000	19.26	0.000048059	16.36	0.0000286953
	800	50	0.5	10	8000	22.89	0.0000484262	16.37	0.0000284147
	1000	50	0.5	10	10000	28.4	0.0000484063	16.37	0.0000288804
Random Algorithm [4]	**100**	50	0.5	10	63	9.06	0.0012812332	16.36	0.0012514842
	200	50	0.5	10	125	17.89	0.001328366	16.35	0.0012859584
	400	50	0.5	10	291	33.17	0.0013339178	16.36	0.0013279629
	600	50	0.5	10	416	49.1	0.0013456784	16.35	0.0012898299
	800	50	0.5	10	561	65.31	0.0013478387	16.36	0.0013304604
	1000	50	0.5	10	722	82.06	0.0013438138	16.36	0.001292203
Simulated Annealing Algorithm [7]	**100**	50	0.5	10	94	5.64	0.0015682236	16.36	0.0015306442
	200	50	0.5	10	205	10.61	0.0014757251	16.35	0.001466698
	400	50	0.5	10	388	20.25	0.0015567327	16.35	0.0015512228
	600	50	0.5	10	564	30.34	0.0015493265	16.36	0.0015167352
	800	50	0.5	10	698	41.97	0.0015714664	16.36	0.001516793
	1000	50	0.5	10	909	50.02	0.0015781129	16.36	0.001550871

0.5. This experiment determines the effect of the number of swarms required for extraction of the operational process of the algorithm.

1) LARGE QUESTION BANK

When changing the number of swarms (changing the number of tests created by single run), the SA algorithm has a higher success rate than Random but does not guarantee to find the correct number of solutions as a requirement. Also, the PSO methods always guarantees a 100% success rate, where sequential PSO is more time-consuming than parallel PSO or parallel migration PSO. Parallel migration PSO has better solution diversity than parallel PSO in Table 5.

2) SMALL QUESTION BANK

With the small search space, PSO methods are always more efficient than Random or SA. PSO methods also find 100% successful solutions in Table 6, which shows that increasing the number of swarms in the algorithm does not decrease the quality of the search results for the bank with a large number of questions. However, the solution diversity of all algorithms is not good as with the large question bank, but is still within

TABLE 7. Experimental results of changing difficulty level requirements in the large question bank.

Algorithms	Difficult level	The number of individuals	The number of swarms	Number of runs	Successful solutions	Average runtime for extracting tests (second)	Average fitness	Average duplicate (%)	Standard deviation
Sequential PSO	**0.3**	50	100	10	1000	92.57	0.0000492182	1.50	0.0000293427
	0.4	50	100	10	1000	51.45	0.0000480883	1.04	0.0000291715
	0.5	50	100	10	1000	22	0.0000490234	0.90	0.0000288239
	0.6	50	100	10	1000	50.11	0.000049519	1.03	0.0000295357
	0.7	50	100	10	1000	91.24	0.0000467931	1.48	0.000029086
Parallel PSO	**0.3**	50	100	10	1000	32.62	0.0000489596	1.51	0.0000289335
	0.4	50	100	10	1000	22.89	0.0000465526	1.05	0.0000285598
	0.5	50	100	10	1000	16.3	0.0000491724	0.90	0.0000285169
	0.6	50	100	10	1000	21.89	0.0000490775	1.03	0.0000288687
	0.7	50	100	10	1000	33.21	0.000047343	1.48	0.0000286973
Parallel Migration PSO	**0.3**	50	100	10	1000	29.76	0.0000489974	1.51	0.0000289427
	0.4	50	100	10	1000	22.27	0.0000486969	1.04	0.0000289297
	0.5	50	100	10	1000	16.06	0.0000472455	0.90	0.0000284014
	0.6	50	100	10	1000	21.21	0.0000470135	1.03	0.0000289434
	0.7	50	100	10	1000	29.77	0.0000473716	1.49	0.0000287463
Random Algorithm [4]	**0.3**	50	100	10	-	-	-	-	-
	0.4	50	100	10	-	-	-	-	-
	0.5	50	100	10	151	56.81	0.0006396715	0.90	0.0006429522
	0.6	50	100	10	-	-	-	-	-
	0.7	50	100	10	-	-	-	-	-
Simulated Annealing Algorithm [7]	**0.3**	50	100	10	**0**	67.88	0.1437704138	0.94	0.0123036347
	0.4	50	100	10	**0**	67.57	0.0430468291	0.94	0.0121575194
	0.5	50	100	10	177	61.38	0.0009304813	0.91	0.0009499761
	0.6	50	100	10	**0**	67	0.039578313	0.95	0.0121070314
	0.7	50	100	10	**0**	67.23	0.1391715678	0.95	0.0122603345

the allowed threshold. Parallel migration PSO has the better runtime and standard deviation than parallel PSO.

We show that the processing time increases as the number of swarms increases, and the standard deviation of both PSO methods is unaffected by changing swarm numbers. However, with question banks that are small, we can see a difference in the search capability of the methods. The processing time of the migration method is always better than that of the non-migratory one, and the standard deviation is significantly improved with better search results. The experimental results show that the average rate of change depends on the number of swarms. PSO methods are stable at 100%, and these demonstrate the diversity of solutions. The average rate does not depend that much on the number of swarms of the algorithm.

D. EVALUATION OF DIFFICULTY LEVEL REQUIREMENTS

In this section, we present evaluations for the algorithms regarding the stability when level of difficulty requirements is varied from 0.3 to 0.7. This experiment aims to evaluate the effect of difficulty level requirements of the algorithms.

1) LARGE QUESTION BANK

Based on the experimental results, we could have a general idea that when difficulty level requirements change this will affect the search capability of the algorithms. In particular, Random and SA only found the solution when the difficulty was 0.5, but the success rate was low. And the PSO methods show that the closer the difficulty level to the margins the more difficult it is to extract tests, so the processing time

TABLE 8. Experimental results of changing the difficulty level requirements in the small question bank.

Algorithms	Difficult level	The number of individuals	The number of swarms	Number of runs	Successful solutions	Average runtime for extracting tests (second)	Average fitness	Average duplicate (%)	Standard deviation
Sequential PSO	**0.3**	50	100	10	38	299.48	0.0164667586	39.85	0.0106887404
	0.4	50	100	10	1000	60.37	0.0000484941	22.75	0.0000287457
	0.5	50	100	10	1000	9.67	0.0000474847	16.36	0.0000286675
	0.6	50	100	10	1000	33.25	0.0000494227	18.04	0.0000280752
	0.7	50	100	10	998	82.74	0.0000584815	28.34	0.0003273015
Parallel PSO	**0.3**	50	100	10	43	78.04	0.0169418951	39.67	0.0109329888
	0.4	50	100	10	1000	17.03	0.000047108	22.71	0.0000288838
	0.5	50	100	10	1000	3.23	0.0000472743	16.39	0.0000290218
	0.6	50	100	10	1000	9.32	0.0000466661	18.02	0.0000278257
	0.7	50	100	10	1000	24.3	0.0000474793	28.38	0.0000292347
Parallel Migration PSO	**0.3**	50	100	10	41	66.1	0.0169272424	39.77	0.0109041026
	0.4	50	100	10	1000	13.31	0.000047068	22.77	0.0000283605
	0.5	50	100	10	1000	2.76	0.0000481289	16.36	0.0000293622
	0.6	50	100	10	1000	7.68	0.0000475494	18.02	0.0000282301
	0.7	50	100	10	998	18.2	0.0000477812	28.34	0.0000485107
Random Algorithm [4]	**0.3**	50	100	10	-	-	-	-	-
	0.4	50	100	10	-	-	-	-	-
	0.5	50	100	10	78	4.61	0.0013501261	16.34	0.0012932811
	0.6	50	100	10	-	-	-	-	-
	0.7	50	100	10	-	-	-	-	-
Simulated Annealing Algorithm [7]	**0.3**	50	100	10	0	5.68	0.1876096688	16.66	0.0082875696
	0.4	50	100	10	0	5.54	0.0878377059	16.65	0.0081876554
	0.5	50	100	10	104	5.21	0.0015220039	16.34	0.0015335744
	0.6	50	100	10	0	5.36	0.0352358466	16.64	0.008081353
	0.7	50	100	10	0	5.38	0.1348998311	16.64	0.0080307427

for generation and the standard deviation is longer. However, the results of the migration of multiple swarm PSO methods demonstrate that the processing speed and diversity of the solutions are better than for non-migratory PSO, as show in Table 7.

2) SMALL QUESTION BANK

With changing the difficulty from 0.3 to 0.7 in Table 8, PSO methods have a lower success rate compared with that seen in the large question banks and the difficulty level is 0.3, the average duplicate is over the threshold (∼10%). Besides, the time needed to find the solution alternates between levels of difficulty. However, PSO methods are very feasible. In contrast, Random and SA are not effective in this case. Parallel migration PSO also shows faster processing time than non-migratory methods. Especially at the difficulty levels of 0.5 and 0.6, both algorithms find the correct solution to the problem, which again proves that when the difficulty level requirements are varied it always affects the search capability of the algorithms.

The above experiments show that there are some limitations if we only use PSO to solve optimization problems. For example, in a few special cases the PSO is only good for global optimization but not good for local optimization. To overcome this limitation, we proposed the application of a parallel multi-swarm algorithm based on the Lewis theory of migration. The theory is based on a basic concept from economics with some modifications for the PSO algorithm to help execute multiple swarms and obtain multiple solutions. We have proven the effectiveness of the PSO in parallel migration in the problem of extracting multiple-choice tests with a difficulty level predefined by the user. Besides, we have evaluated and compared the efficiency of the proposed algorithm and other algorithms from previous works,

such as Random [4] and SA [7]. Our proposed algorithms are always highly effective based on the evaluation criteria for the change in the number of individuals in the swarm, change in the number of swarms, and change in the difficulty of the test.

V. CONCLUSION AND FUTURE STUDIES

In this paper, we present an approach to extract the abundance of tests with equivalent levels of difficulty and approximate the specific difficulty level requirement given by the user based on question banks and parallel multi-swarm migration in PSO algorithm. Our approach performs better than the other techniques, such as Simulate Annealing [7], Random [4], etc., in terms of the execution time and quality of the tests. The experimental results are compared with those from PSO methods such as sequential PSO, parallel PSO and parallel migration PSO and assessed in light of several essential criteria, including number of successful solutions, accuracy, standard deviation, search speed, and number of questions overlapping between the exam questions, as well as for changing the search space (large and small question banks), changing the number of individuals, changing the number of groups (number of questions), and changing the difficulty requirements of the test. The results demonstrated that the developed algorithm produces solutions of good quality (with optimal and near-optimal solutions) in a short amount of computing time.

Future studies may focus on investigating the use of a hybrid approach [31], [34] to solve other NP-hard and combinatorial optimization problems, which a focus on fine-tuning the PSO parameters by using some of adaptive strategies. Additionally, we will extend our problem to provide feedback to instructors from multiple choice data, such as using fuzzy theory by Le and Fujita [35], and PSO with local search algorithms for mining association rules to compute the difficulty levels of questions.

REFERENCES

[1] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. Hoboken, NJ, USA: Wiley, 2005.

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

[3] C. J. A. B. Filho, F. B. de Lima Neto, A. J. C. C. Lins, A. I. S. Nascimento, and M. P. Lima, "A novel search algorithm based on fish school behavior," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2008, pp. 2646–2651.

[4] K. Naik, S. Sule, S. Jadhav, and S. Pandey, "Automatic question paper generation system using randomization algorithm," *Int. J. Eng. Tech. Res.*, vol. 2, no. 12, pp. 192–194, 2014.

[5] F. Kiran, H. Gopal, and A. Dalvi, "Automatic question paper generator system," *Int. J. Comput. Appl.*, vol. 166, no. 10, pp. 42–47, May 2017.

[6] A. M. Marks, "Random question sequencing in computer-based testing (CBT) assessments and its effect on individual student performance," Ph.D. dissertation, Fac. Educ., Univ. Pretoria, Pretoria, South Africa, 2007.

[7] P. Lu, X. Cong, and D. Zhou, "The research on Web-based testing environment using simulated annealing algorithm," *Sci. World J.*, vol. 2014, pp. 1–12, Jan. 2014.

[8] F. Ting, J. H. Wei, C. T. Kim, and Q. Tian, "Question classification for E-learning by artificial neural network," in *Proc. 4th Int. Conf. Inf., Commun. Signal Process.*, 2003, pp. 1575–1761.

[9] G.-J. Hwang, P.-Y. Yin, and S.-H. Yeh, "A tabu search approach to generating test sheets for multiple assessment criteria," *IEEE Trans. Educ.*, vol. 49, no. 1, pp. 88–97, Feb. 2006.

[10] N. S. B. M. Jamail and A. B. M. Sultan, "Shuffling algorithms for automatic generator question paper system," *Comput. Inf. Sci.*, vol. 3, no. 2, pp. 244–248, Apr. 2010.

[11] D. D. Rasika and M. Z. Shaikh, "Automatic test paper generator with shuffling algorithm," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 4, pp. 2176–2180, Feb. 2016.

[12] M. Yildirim, "Heuristic optimization methods for generating test from a question bank," in *Proc. Mex. Int. Conf. Artif. Intell.* Berlin, Germany: Springer, 2007, pp. 1218–1229.

[13] M. Yildirim, "A genetic algorithm for generating test from a question bank," *Comput. Appl. Eng. Educ.*, vol. 18, no. 2, pp. 298–305, 2010.

[14] T. F. Ho, P. Y. Yin, G. J. Hwang, S. J. Shyu, and Y. N. Yean, "Multi-objective parallel test-sheet composition using enhanced particle swarm optimization," *Educ. Technol. Soc.*, vol. 12, no. 4, pp. 193–206, 2009.

[15] C. Zhang and J. Zhang, "Research on test paper auto-generating based on improved particle swarm optimization," in *Proc. 7th Int. Symp. Parallel Archit., Algorithms Program. (PAAP)*, Dec. 2015, pp. 92–96.

[16] C. Zhiyun, W. Yong, and B. Yue, "Personalized question bank research based on particle swarm optimization," in *Proc. Int. Conf. Math., Big Data Anal. Simulation Modeling (MBDASM)*, vol. 92, 2019, pp. 111–114.

[17] S.-C. Cheng, Y.-T. Lin, and Y.-M. Huang, "Dynamic question generation system for Web-based testing using particle swarm optimization," *Expert Syst. Appl.*, vol. 36, no. 1, pp. 616–624, Jan. 2009.

[18] T. Bui, T. Nguyen, B. Vo, N. Nguyen, W. Pedrycz, and V. Snasel, "Application of particle swarm optimization to create multiple-choice tests," *J. Inf. Sci. Eng.*, vol. 34, no. 6, pp. 1405–1423, 2018.

[19] T. Nguyen, T. Bui, and B. Vo, "Multi-swarm single-objective particle swarm optimization to extract multiple-choice tests," *Vietnam J. Comput. Sci.*, vol. 6, no. 2, pp. 147–161, May 2019.

[20] Ş. Gülcü and H. Kodaz, "Effects of the different migration periods on parallel multi-swarm PSO," *Comput. Sci. Inf. Technol.*, vol. 6, no. 6, pp. 13–22, 2016.

[21] Ş. Gülcü and H. Kodaz, "Effects of the number of swarms on parallel multi-swarm PSO," *Int. J. Comput., Commun. Instrum. Eng.*, vol. 3, no. 1, pp. 201–205, 2016.

[22] S.-K.-S. Fan and J.-M. Chang, "A parallel particle swarm optimization algorithm for multi-objective optimization problems," *Eng. Optim.*, vol. 41, no. 7, pp. 673–697, Jul. 2009.

[23] S. Katiyar, "A comparative study of genetic algorithm and the particle swarm optimization," *Int. J. Technol.*, vol. 2, no. 2, pp. 21–24, 2015.

[24] P. Kromer, J. Platos, and V. Snasel, "A brief survey of advances in particle swarm optimization on graphic processing units," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, 2013, pp. 182–188.

[25] W. A. Lewis, "Economic development with unlimited supplies of labour," *Manchester School*, vol. 22, no. 2, pp. 139–191, May 1954.

[26] B. Niu, Y. Zhu, X. He, and H. Wu, "MCPSO: A multi-swarm cooperative particle swarm optimizer," *Appl. Math. Comput.*, vol. 185, no. 2, pp. 1050–1062, Feb. 2007.

[27] D. D. Patil and B. D. Dangewar, "Multi-objective particle swarm optimization (MOPSO) based on Pareto dominance approach," *Int. J. Comput. Appl.*, vol. 107, no. 4, pp. 13–15, Dec. 2014.

[28] S. Shabir and R. Singla, "A comparative study of genetic algorithm and the particle swarm optimization," *Int. J. Electr. Eng.*, vol. 9, no. 2, pp. 215–223, 2016.

[29] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, Jan. 2018.

[30] H. Zhou, M. Song, and W. Pedrycz, "A comparative study of improved GA and PSO in solving multiple traveling salesmen problem," *Appl. Soft Comput.*, vol. 64, pp. 564–580, Mar. 2018.

[31] F. Zou, D. Chen, R. Lu, and P. Wang, "Hierarchical multi-swarm cooperative teaching–learning-based optimization for global optimization," *Soft Comput.*, vol. 21, no. 23, pp. 6983–7004, Dec. 2017.

[32] L. Marques and A. T. Almeida, "Finding odours across large search spaces: A particle swarm-based approach," in *Climbing and Walking Robots*. Berlin, Germany: Springer, 2005, pp. 419–426.

[33] Ł. Strąk, R. Skinderowicz, and U. Boryczka, "Adjustability of a discrete particle swarm optimization for the dynamic TSP," *Soft Comput.*, vol. 22, no. 22, pp. 7633–7648, Nov. 2018.

[34] S. Sedarous, S. M. El-Gokhy, and E. Sallam, "Multi-swarm multi-objective optimization based on a hybrid strategy," *Alexandria Eng. J.*, vol. 57, no. 3, pp. 1619–1629, Sep. 2018.

[35] L. H. Son and H. Fujita, "Neural-fuzzy with representative sets for prediction of student performance," *Int. J. Speech Technol.*, vol. 49, no. 1, pp. 172–187, Jan. 2019.

[36] E. Ridge and D. Kudenko, "Tuning an algorithm using design of experiments," in *Experimental Methods for the Analysis of Optimization Algorithms*. Berlin, Germany: Springer, 2010, pp. 265–286.
[37] J. H. Holland, *Adaptation in Natural and Artificial Systems*, 2nd ed. Ann Arbor, MI, USA: Univ. Michigan Press, 1992.
[38] S. Ayesha, T. Mustafa, A. Sattar, and M. I. Khan, "Data mining model for higher education system," *J. Sci. Res.*, vol. 43, pp. 24–29, Jan. 2010.

TRAM NGUYEN received the M.Sc. degree in computer science from the University of Information Technology—Viet Nam National University HCMC, in 2013. She is currently pursuing the Ph.D. degree with the VŠB—Technical University of Ostrava, Czech Republic. Her research interests include evolutionary algorithm, particle swarm optimization, genetic algorithm, grid computing, and parallel computing.

LOAN T. T. NGUYEN received the B.Sc. degree in computer science from the University of Science, Ho Chi Minh City, Vietnam, in 2002, the M.Sc. degree in computer science from Vietnam National University, Ho Chi Minh City, in 2008, and the Ph.D. degree in computer science from the Wroclaw University of Science and Technology, Poland, in 2015. From October 2016 to September 2017, she was an ERCIM Postdoctoral Researcher with the University of Warsaw, Poland. From March to March 2017, she was a Visiting Researcher with NTNU, Norway. Her research interests include association rules, classification, and mining in incremental databases.

TOAN BUI is a Teaching Assistant with the Faculty of Information Technology, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam. His research interests include evolutionary algorithm, particle swarm optimization, genetic algorithm, grid computing, and parallel computing.

HO DAC LOC received the Ph.D. degree from the Moscow Power Engineering Institute, Russia, in 2002. He is currently a Professor and the President of the Ho Chi Minh City University of Technology, Vietnam. His research interests include metaheuristic algorithms, power engineering, and cloud computing.

WITOLD PEDRYCZ (Life Fellow, IEEE) is currently a Professor and a Canada Research Chair (CRC) in computational intelligence with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is also with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. He holds an appointment of special professorship with the School of Computer Science, University of Nottingham, U.K. He has published numerous papers in this area. He is also an author of 15 research monographs covering various aspects of computational intelligence, data mining, and software engineering. His main research interests include computational intelligence, fuzzy modeling and granular computing, knowledge discovery and data mining, fuzzy control, pattern recognition, knowledge-based neural net-works, relational computing, and software engineering. He was elected as a Foreign Member of the Polish Academy of Sciences, in 2009. He was elected as a Fellow of the Royal Society of Canada, in 2012. He has been a member of numerous program committees of IEEE conferences in the area of fuzzy sets and neurocomputing. He was a recipient of the IEEE Canada Computer Engineering Medal 2008. He received a prestigious Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Council, in 2007, the Cajastur Prize for Soft Computing from the European Centre for Soft Computing for pioneering and multifaceted contributions to Granular Computing, in 2009, the Killam Prize, in 2013, and the Fuzzy Pioneer Award 2013 from the IEEE Computational Intelligence Society, in 2013.

VACLAV SNASEL (Senior Member, IEEE) received the Master of Science degree from Palacky University, Olomouc, Czech Republic, and the Ph.D. degree in algebra and geometry from Masaryk University, Brno, Czech Republic.

Since 2001, he has been a Visiting Scientist with the Institute of Computer Science, Academy of Sciences, Czech Republic. Since 2003, he has been the Vice-Dean for Research and Science with the Faculty of Electrical Engineering and Computer Science, VSB—Technical University of Ostrava, Czech Republic. Since 2006, he has been a Full Professor. Before turning into a full-time academic, he was working with industrial company where he was involved in different industrial research and development projects for nearly eight years. He has research and development experience over 25 years in the Industry and Academia. He works in a multi-disciplinary environment involving artificial intelligence, multidimensional data indexing, conceptual lattice, information retrieval, semantic web, knowledge management, data compression, machine intelligence, neural networks, web intelligence, data mining, and applied to various real-world problems. He has given more than ten plenary lectures and conference tutorials in these areas. He has authored/coauthored several refereed journal/conference papers and book chapters. He has published more than 400 articles (147 is recorded at Web of Science). He has supervised many Ph.D. students from Czech Republic, Jordan, Yemen, Slovakia, Ukraine, and Vietnam. He is a member of ACM, AMS, and SIAM. He is the Editor-in-Chief of two journals. He also serves the editorial board of some reputed International journals. He is actively involved in the International Conference on Computational Aspects of Social Networks (CASoN), Computer Information Systems and Industrial Management (CISIM), and Evolutionary Techniques in Data Processing (ETID) series of International conferences.

BAY VO received the Ph.D. degree in computer science from the University of Science, Vietnam National University of Ho Chi Minh, in 2011. He is currently an Associate Professor and the Dean of Faculty. His research interests include association rule mining, classification, incremental mining, distributed databases, and privacy preserving in data mining. He is a member of the Review Board of the International Journal of Applied Intelligence. He also served as a Co-Chair of several special sessions, such as ICCCI 2012; ACIIDS 2013, 2014, 2015, 2016; KSE 2013, 2014, and SMC 2015. He serves as an Associate Editor of the *ICIC Express-Letters, Part B: Applications*. He is an Editor of the *International Journal of Engineering and Technology Innovation*. He serves as a reviewer of many international journals, such as the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, *KAIS*, *ESWA*, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, *Information Sciences*, *Knowledge Based Systems*, *Soft Computing*, *JISE*, *PLOS ONE*, IEEE ACCESS, and so on.