Hindawi
Complexity
Volume 2018, Article ID 4730957, 14 pages
https://doi.org/10.1155/2018/4730957

Research Article

Multipopulation Management in Evolutionary Algorithms and Application to Complex Warehouse Scheduling Problems

Yadong Yu,[1] Haiping Ma ,[1] Mei Yu,[1] Sengang Ye,[1] and Xiaolei Chen[2]

[1]*Department of Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang, China*
[2]*School of Automation, Chongqing University of Posts and Telecommunications, Chongqing, China*

Correspondence should be addressed to Haiping Ma; mhping1981@126.com

Received 3 February 2018; Accepted 18 March 2018; Published 26 April 2018

Academic Editor: Liang Hu

Multipopulation is an effective optimization strategy which is often used in evolutionary algorithms (EAs) to improve optimization performance. However, it is of remarkable difficulty to determine the number of subpopulations during the evolution process for a given problem, which may significantly affect optimization ability of EAs. This paper proposes a simple multipopulation management strategy to dynamically adjust the subpopulation number in different evolution phases throughout the evolution. The proposed method makes use of individual distances in the same subpopulation as well as the population distances between multiple subpopulations to determine the subpopulation number, which is substantial in maintaining population diversity and enhancing the exploration ability. Furthermore, the proposed multipopulation management strategy is embedded into popular EAs to solve real-world complex automated warehouse scheduling problems. Experimental results show that the proposed multipopulation EAs can easily be implemented and outperform other regular single-population algorithms to a large extent.

1. Introduction

Evolutionary algorithms (EAs) are fast and robust computation methods for global optimization and have been widely applied in solving numerous real-world problems [1–5]. In recent years, the concept of multipopulation is frequently discussed and used to improve the optimization performance of EAs. In this regard, firstly, the original population is divided into several small subpopulations for special purposes such as large-scale problems and dynamic optimization problems. Then, particular evolution mechanisms and operations, for example, crossover and mutation for genetic algorithms (GAs), are executed. The purpose of multipopulation is to maintain population diversity and enhance exploration ability, which is the crucial factor to avoid premature convergence when handling optimization problems.

Existing studies on multipopulation demonstrate that such strategy has become one of the most effective methods to enhance EA performance [6, 7]. The key reasons for this are categorized as follows: (1) it divides the overall population into multiple subpopulations, in which the population diversity can be maintained due to the fact that different subpopulations can be located in different search domains; (2) it is able to search different areas simultaneously, leading the separated populations to rapidly find the optimal solutions; and (3) various population-based EAs can be fast and easily integrated within multipopulation methods. In the research work carried out by Chang [8], a modified particle swarm optimization (PSO) with multiple subpopulations was utilized for multimodal function optimization problems. Simulation results on complex multimodal functions showed that the global and local system solutions were solved by these best particles of subpopulations. Nseef et al. [9] proposed an adaptive multipopulation artificial bee colony (ABC) algorithm for dynamic optimization problems, where multiple subpopulations were used to cope with dynamic changes and to maintain the diversity. Experimental results showed that the proposed multipopulation ABC was superior to regular ABC on all tested instances. In the research work carried out by Wu et al. [10], differential evolution (DE) with multipopulation-based ensemble of mutation strategies was used for optimizing benchmark functions, where each subpopulation performed a mutation strategy. Experimental

comparisons showed the competitive performance of the proposed method. A hybrid multipopulation GA was employed in [11] for the dynamic facility layout problem, where the entire solution space was separated into different parts and each subpopulation represented a separate part. Simulation results showed that the proposed algorithm enjoyed the superiority over other algorithms. Niu et al. [12] proposed a symbiosis-based alternative learning multi-swarm PSO algorithm, where the communication between subpopulations used a learning method to select one example out of the center position, the local best position, and the historical best position including the experience of the internal and external multiple subpopulations, to keep the diversity of the population. The experimental results exhibited better performance in terms of the convergence speed and optimality. Xiao et al. [13] presented a novel multipopulation coevolution immune optimization algorithm (IOA) for most of the existing multimodal benchmarks, where coevolution of three subpopulations was promoted through a self-adjusted clone operator to enhance exploration and exploitation. The authors proved that their method outperformed three known immune algorithms and several other EAs. The introduction of external archiving into a multipopulation harmony search (HS) algorithm to solve dynamic optimization problems was presented by Turky and Abdullah [14]. The results on moving peak benchmarks showed that their modified version was better than the original harmony search algorithms. A multipopulation cooperative bat algorithm (BA) was used in [15] for an artificial neural network model, which mainly depended on the connection weights and network structure. Experimental results showed that there was a significant improvement by applying the proposed algorithm to all the test cases. Ozsoydan and Baykasoğlu [16] employed a multipopulation firefly algorithm (FA) to tackle dynamic optimization problems. The experiments on moving peak benchmarks showed that the proposed algorithm significantly improved system performance. Mauša and Galinac Grbac [17] proposed a coevolutionary multipopulation genetic programming (GP) which combined colonization and migration with three ensemble selection strategies for classification in software defect prediction. Computational results demonstrated the efficiency of the proposed method.

Although multipopulation methods have shown success for solving optimization problems, most of them use a preset constant number of subpopulations during optimization process [18]. The subpopulation number has an important impact on performance of multipopulation EAs, because it is related to the difficulty of the problem, which is not known in advance. For a given problem, each EA may have different subpopulation numbers during different phases of the search process. For example, an algorithm with many subpopulations may have effective search ability during the initial phase of the optimization process, whereas the algorithm with a few subpopulations may have better search ability during the later phase of the optimization process. Therefore, it is of potential to dynamically manage the number of subpopulations during evolution process, based on the difficulty of the problem. It may result in outstanding results without the essential use of dedicated evolution operators. Nonetheless in most of multipopulation EA literature [19, 20], the dynamic management of subpopulation number is rarely mentioned.

In order to address this issue, this paper proposes a multipopulation management strategy to improve EA optimization performance. The proposed method uses some simple rules to dynamically manage the subpopulation number to maintain population diversity. To verify its effectiveness, the proposed multipopulation management strategy is embedded into various popular algorithms to construct multipopulation EAs. Then they are tested on a set of CEC benchmark functions and further applied to real-world complex automated warehouse scheduling problems. Experimental results show that the proposed multipopulation management strategy can help EAs to obtain excellent results and outperform some state-of-the-art single-population algorithms in the literature.

The following are the original contributions of this paper: (1) a dynamic management strategy of the subpopulation number is proposed to boost population diversity while preserving simplicity for EAs; (2) the proposed multipopulation management strategy is embedded into several popular EAs, including the stud genetic algorithm (SGA), population-based incremental learning (PBIL), self-adaptive differential evolution (SaDE), and standard particle swarm optimization (PSO2011); (3) the optimization ability of these multipopulation EAs is investigated on a set of benchmark functions, and further they are applied to solve real-world automated warehouse scheduling problems.

The remainder of the paper is organized as follows. Section 2 gives detail descriptions of multipopulation management strategy. Section 3 discusses the integration of multipopulation management strategy with popular EAs. Section 4 compares the performance of several EAs in conjunction with multipopulation management strategy on benchmark functions and complex warehouse scheduling problems. Section 5 provides conclusions and suggestions for future research.

2. Multipopulation Management Strategy

This section presents the basic challenges for multipopulation EAs, as well as our proposed dynamic management for the subpopulation number.

2.1. Challenges for Multipopulation Methods. In EAs, diversity refers to differences among candidate solutions. As mentioned in most of EA literature, evolution progress lies fundamentally on the existence of variation of population, and the high diversity of a population greatly contributes to EA performance. Diversity loss often results in premature convergence, because EAs find themselves trapped in local optimal solutions and lack population diversity to escape. So the crucial point for EAs solving optimization problems is how to maintain population diversity. In recent years, multipopulation methods are introduced into EAs and become one of the most successful methods to improve optimization performance, because such methods deal with candidate solutions scattering over the entire search space. This feature

helps population-based EAs to quickly achieve the global optimal solutions.

To make multipopulation methods more efficient, several crucial challenging issues in algorithm design need to be addressed. The first question is how to determine the number of subpopulations. If too many subpopulations distribute in the problem, this may waste the limited computational resources. However, too small number of subpopulations may lead to limited effect of the multipopulation strategy. The second question is how to determine the search area of each subpopulation. If the search area of a subpopulation is too small, there is a potential problem that the small isolated subpopulation may converge to a local optimal solution. In this case, the diversity will lose and the algorithm can hardly make any progress. On the contrary, if the search area of a subpopulation is too large, it is almost equal to the search area of the original population, and it is very hard to obtain better optimization performance compared with the regular algorithms. The third question is how to communicate between subpopulations. Many researchers believe that communication between subpopulations is very helpful during optimization because information can be shared among subpopulations and, hence, this will accelerate the search process and promising solutions may be found as well. In some current studies, the communication between subpopulations is controlled by four parameters: (i) a communication rate that defines the number of solutions in a subpopulation to be sent to other subpopulations; (ii) a communication policy that determines which solutions are to be replaced by ones from other subpopulations; (iii) a communication interval that sets up the frequency for executing communication; and (iv) a connection topology that defines how to connect between subpopulations.

For these challenges, most existing multipopulation methods just use predefined values, which are based on empirical experience, to determine the parameter setting of subpopulations. Some other studies assume that some information of optimization problems is known. In these cases, problem information can be used to guide the configuration of multipopulation parameters. However, for most of the cases, we need to deeply explore these challenging issues to develop excellent multipopulation EAs.

For the first challenge mentioned above, it is a good method that we use the appropriate number of subpopulations to maintain population diversity. This issue includes two ways. The first way is to use a fixed number of subpopulations. Most of current multipopulation methods fall into this group. The advantage of this way is that it can be implemented simply, and we only need to create a fixed number of subpopulations in advance for the optimization problem. Generally speaking, the more the peaks are in the fitness functions, the more the subpopulations are needed. However, it is not efficient to obtain the number of peaks of fitness functions for practice problems. In addition, the distribution and shape of fitness functions may also play a role in configuring the subpopulation number. The second way is to use a variable number of subpopulations. It is a difficult problem when the number of subpopulations is increased or decreased. To maintain population diversity, the subpopulation number might be different at different states during evolution process. For example, in the early stages, it needs a large number of subpopulations because candidate solutions can scatter over the entire search space, which leads to high population diversity. But, in the later stages, a small number of subpopulations are in favor of reducing diversity to fast converge to global best solutions. So the dynamic change of the subpopulation number should be in line with the population diversity. In many previous studies [21, 22], it often splits off from a main population into multiple parts to increase the subpopulation number and merges a set of small subpopulations into a main population to decrease the subpopulation number. The common characteristic of these methods is that they often adopt a specific cluster method to complete these operations. Therefore, additional knowledge about clustering is needed meanwhile increasing the complexity of algorithms, which is not desirable for solving some complex problems with the limitation of computational resource.

Undoubtedly, it is a good idea to dynamically manage the number of subpopulations during evolution process, without introducing additional complex mechanisms and considering the difficulty of the optimization problems in advance.

2.2. Ways to Manage Subpopulation Number. This part introduces a simple and effective multipopulation management strategy, to dynamically increase or decrease the subpopulation number. Compared with other multipopulation methods, the proposed method highlights the challenge of determining the number of subpopulations.

In the ways to manage the subpopulation number, four basic rules are considered:

(1) The maximum number of subpopulations is limited in order to prevent computational burden on too many coevolving subpopulations.

(2) The subpopulation number decreases, when we merge the same subpopulations or delete the existing subpopulations.

(3) The subpopulation number increases, when we create some new subpopulations or divide the existing subpopulation.

(4) The interaction of subpopulations is not considered because it is so smart that it affects the investigation on multipopulation management strategy.

Based on the above basic rules, two open problems need to be solved in the proposed multipopulation management strategy. First, what is the best condition to increase or decrease the subpopulation number? Before answering this question, we firstly consider that the purpose of multipopulation methods is to maintain population diversity. So it is an effective method to build the relationship between population diversity and control condition of the subpopulation number. Quantitatively, we need some simple approaches to measure diversity. Euclidean distance is probably the most widely used

type of diversity measure nowadays. Euclidian distance d_E between two solutions s_i and s_j can be calculated as

$$d_E\left(s_i, s_j\right) = \sqrt{\sum_{l=1}^{n}\left(s_{il} - s_{jl}\right)^2}, \quad (1)$$

where s_{il} and s_{jl} denote the lth solution variables in the solutions s_i and s_j, respectively, and n is the number of solution variables.

Equation (1) is used to measure individual distance between two solutions. In the multipopulation method, we need to further consider population distance between two subpopulations. To calculate population distance, the concept of average population is introduced, which is a vector of average value of all solutions in the same population. We use $\bar{s}$ to represent average population, which is calculated as follows:

$$\bar{s} = \frac{\sum_{i=1}^{N} s_i}{N}, \quad (2)$$

where N is the size of the population.

The corresponding standard deviation of average population is calculated as

$$\sigma_s = \sqrt{\frac{\sum_{i=1}^{N}\left(s_i - \bar{s}\right)}{N}}. \quad (3)$$

Then, we define population distance $D_E(P_i, P_j)$ between two subpopulations P_i and P_j, which is represented as

$$D_E\left(P_i, P_j\right) = d_E\left(\bar{s}, \bar{t}\right) - k\left[d_E\left(\bar{s}, \sigma_s\right) + d_E\left(\bar{t}, \sigma_t\right)\right], \quad (4)$$

where $d_E(\bar{s}, \bar{t})$, $d_E(\bar{s}, \sigma_s)$, and $d_E(\bar{t}, \sigma_t)$ are Euclidean distances, $\bar{s}$ and $\bar{t}$ are average populations, and σ_s and σ_t are standard deviations for subpopulations P_i and P_j, respectively. Equation (4) denotes the fact that population distance between two subpopulations is related to Euclidean distances between their average populations, and solution distribution in each subpopulation. If D_E is small, areas occupied by subpopulations P_i and P_j are overlapping, which means that they have great similarity. The coefficient k is set to 2 on the basis of normal distribution characteristic [23], according to 95% of solutions being located in distance of two standard deviations from average populations.

Next, we consider several diversity cases, and they are defined by the following ways:

(1) As a diversity between individuals in the same subpopulation;

(2) As a diversity between a subpopulation and other subpopulations;

(3) As a diversity between the current subpopulation and its parent subpopulation.

Note that the first case is taken as individual diversity; the second and third cases are taken as population diversity.

Once we have the method of measuring diversity, only a threshold value T is set to decide the condition to increase or decrease the subpopulation number. For individuals in the same subpopulation, if $d_E > T$ for any one of individual distances, we reckon that they are different, and if $d_E < T$ for all individual distances, they are considered to be similar. In the same way, for any two subpopulations, if $D_E > T$, they are regarded as of diversity, and if $D_E < T$, we think that they have similarity. Note that similarity is the opposite of diversity; T can be a parameter by a user or can be an adaptive parameter.

Second, how do we increase or decrease the subpopulation number during evolution process? That is, how do we create new subpopulations, or delete redundant subpopulations? Based on the diversity concepts mentioned above, multipopulation management strategy can be roughly classified into three cases. In the first case, when all individuals are similar in the certain subpopulation, we firstly randomly preserve one of individuals because they are almost the same and then delete this subpopulation. Meanwhile, we create a new subpopulation consisting of the following individuals: 1/3 of individuals are copies of the preserved individual, 1/3 of individuals are from the best individuals in the entire population, and 1/3 of individuals are randomly generated. In the second case, when a subpopulation is similar to other subpopulations within the whole population, we directly delete this subpopulation to save computational resources. In the third case, when a subpopulation is similar to its parent subpopulation, we create a new subpopulation, and it consists of the following individuals: 50% of individuals are copies of the best individual in the latest generation, and the other 50% are generated randomly from the neighborhood of the best individual with the normal distribution. Note that the maximum number of subpopulations is limited to the preset number to prevent too many subpopulations of being involved. Figure 1 shows the multipopulation management strategy during evolution process.

3. Integration with EAs

Now we integrate the proposed multipopulation management strategy with EAs, called multipopulation EAs, and the flowchart is shown in Figure 2. It starts by setting the parameters. It creates a population of candidate solutions and evaluates them. Next, the population of solutions is divided into multiple subpopulations. Each subpopulation runs an EA paradigm to generate its own offspring. Then, subpopulations perform diversity judgement, and, based on diversity levels, it creates or deletes subpopulations. Finally, it checks the stopping condition. In this way, EAs always can dynamically manage the subpopulation number, leading to better performance than the corresponding single-population EAs. The pseudocode of multipopulation EAs is shown in Algorithm 1.

The main steps of the proposed multipopulation EAs are further described in detail below.

Step 1 (set parameters). The main parameters of the proposed multipopulation EAs are initialized. They include the parameters of EA paradigms, the maximum number of iterations, the size of population, the number of initial subpopulations, and the maximum number of subpopulations.

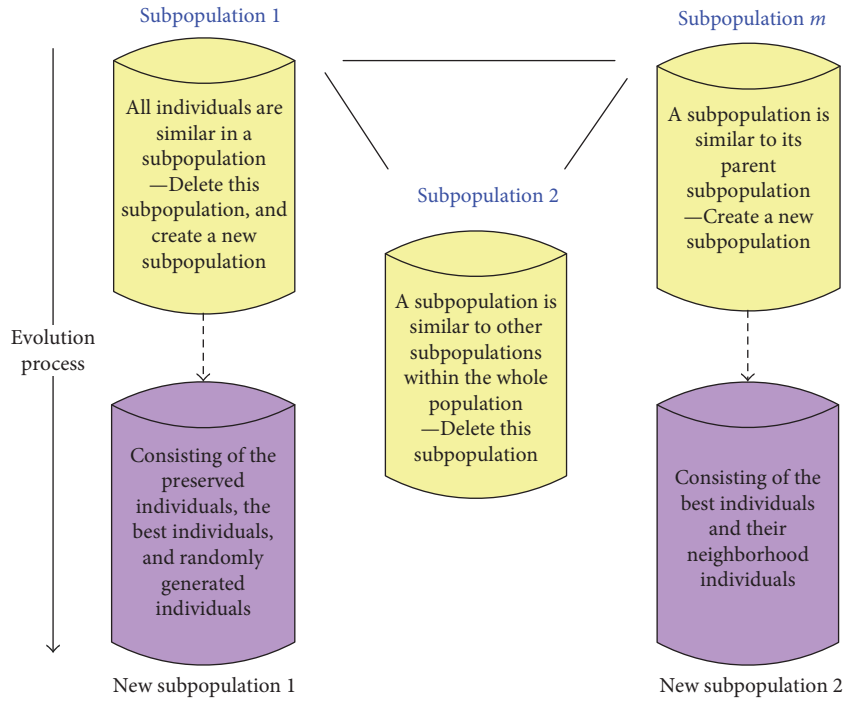

Figure 1: Multipopulation management strategy during evolution process.

Step 2 (initialize the population of solutions). It randomly generates a set of candidate solutions between the lower and upper boundaries.

Step 3 (evaluate the population of solutions). The fitness of the generated solutions is calculated using the objective function.

Step 4 (divide the population). The initial population is divided into multiple subpopulations with the same population size, and each subpopulation is randomly assigned from solutions in the population.

Step 5 (create offspring subpopulations). Different subpopulations are executed independently by EA paradigms to generate their own offspring subpopulations. Note that, for EA paradigms, we can use the same EA for all subpopulations, or we also can use the different EA for each subpopulation.

Step 6 (judge diversity and manage subpopulations). If a subpopulation is similar to other subpopulations, directly delete this subpopulation. If it is similar to its parent subpopulation, create a new subpopulation. If individuals are similar in a subpopulation, firstly delete this subpopulation and then create a new subpopulation. For details see Section 2.2.

Step 7 (evaluate offspring subpopulations and check the stopping condition). If the termination criterion is not met, go to Step 5; otherwise, terminate and output the evaluation results. Here the termination criterion is the maximum number of function evaluations.

4. Experimental Results

In this section the performances of the proposed multipopulation EAs are investigated. Section 4.1 describes the experimental setup, Section 4.2 presents performance comparisons on the 2013 CEC benchmark functions, and Section 4.3 applies the proposed multipopulation EAs to a real-world complex automated warehouse scheduling problem.

4.1. Simulation Setup. The performances of the proposed multipopulation EAs are evaluated on the 28 benchmark functions presented in Table 1, which are from the 2013 Congress on Evolutionary Computation (CEC) [24].

The popular EAs used in this paper include SGA, PBIL, SaDE, and PSO2011. We select SGA because it is an improvement of the basic GA that uses the best individual in each generation for crossover [25]. We select PBIL because it is the most successful variant of estimation of distribution algorithms [26]. We select SaDE because it is one of the most powerful DE algorithms and has demonstrated excellent performance on many problems [27, 28]. We select PSO2011 because it is popular in the literature and contains improvements gained as a result of many years of PSO studies [29, 30].

The next step is to set the parameters of each multipopulation EA. For SGA we use real coding, roulette-wheel selection, single-point crossover with a crossover probability of 1, and a mutation probability of 0.001. For PBIL we use learning rate $\eta = 0.1$, the number of best and worst individuals used to update probabilities in each generation is

(1) Set the parameters of the proposed multi-population EAs
(2) Randomly initialize the entire population
(3) Evaluate the fitness of all candidate solutions in the population
(4) Divide the population into multiple subpopulations
(5) While the halting criterion is not satisfied do
(6) For each subpopulation do
(7) Perform an independent EA to create its own offspring subpopulation
(8) End for
(9) Judge diversity and manage offspring subpopulations, and the detail sees Section 2.2
(10) Evaluate the fitness of all offspring subpopulations
(11) End while

Algorithm 1: Pseudocode of the proposed multipopulation EAs.

$N_{best} = N_{worst} = 5$, and the standard deviation for mutation linearly decreases from 10% of the parameter range at the first generation to 2% of the parameter range at the final generation [26]. The SaDE parameter settings are adapted according to the learning progress [27]: the scaling factor F is randomly sampled from the normal distribution $N(0.5, 0.3)$, and the crossover rate CR follows the normal distribution $N(0.5, 0.1)$. For PSO2011, we use an inertia weight $w = 1/(2 \cdot \log(2))$, a cognitive constant $c_1 = 0.5 + \log(2)$, and a social constant c_2 for neighborhood interaction that is the same as c_1.

In addition, we initially use three subpopulations, and the maximum subpopulation number is set to six. Each subpopulation uses the same EA, and the population size of each subpopulation is 25. For fair comparisons, the population size of single-population EAs is set to 150. We evaluate each function in $D = 30$ dimensions with the function evaluation limit of $D \times 10{,}000$. All algorithms are terminated after the maximum number of function evaluations is reached, or if the objective function error value is below 10^{-6}.

4.2. Performance Comparisons. We simulate each algorithm 25 times on each benchmark, and the results are shown in Tables 2 and 3. The tables show that multipopulation SaDE (M-SaDE) performs best on 18 functions (F4, F6, F7, F9, F10, F11, F12, F13, F14, F15, F16, F18, F19, F20, F21, F22, F26, and F28), and multipopulation PSO2011 (M-PSO2011) performs best on 9 functions (F2, F3, F5, F8, F17, F23, F24, F25, and F27). For function F1, both M-SaDE and M-PSO2011 obtain the global optimal solutions. Furthermore, we briefly consider the types of functions for which the various algorithms are best-suited. Tables 2 and 3 show that the best-performing algorithm on each of the unimodal functions (F1–F5) is always M-PSO2011, and the best-performing algorithm on each of the multimodal functions (F6–F20) is always M-SaDE. We also note from these tables that M-PSO2011 performs as well as M-SaDE on the composition functions (F21–F28). This implies that no single algorithm can be the best for every problem, and, for different optimization problems, different algorithms have their own superiority to obtain the best performance.

Tables 2 and 3 also show that the total performances of multipopulation EAs, including M-SGA, M-PBIL, M-SaDE, and M-PSO2011, are better than their corresponding single-population EAs. This may be due to the proposed multi-population management strategy that enhances population diversity to improve optimization performance.

The average running times of all algorithms are shown in the last rows of Tables 2 and 3. Here MATLAB® is used as the programming language, and the computer is a 2.40 GHz Intel Pentium® 4 CPU with 4 GB of memory. We find that the average running times of the multipopulation EAs are less than their corresponding single-population algorithms. For example, the average running time of multipopulation SGA (M-SGA) is less than single-population SGA (S-SGA). The reason is that multipopulation EAs use multiple parallel subpopulations to reduce computation time with the same total population size as the corresponding single-population algorithms. So multiple subpopulations are also amenable to parallel processing, and they can further reduce computational effort.

In order to further compare the performance of the multipopulation and single-population EAs, we use the Wilcoxon method to test for statistical significance. The Wilcoxon method is a nonparametric statistical test to determine whether differences between groups of data are statistically significant when the assumptions that the differences are independent and identically normally distributed are not satisfied [31–33]. The Wilcoxon test results are shown in Table 4, where the pairs are marked if the difference between each pair of algorithms is statistically significant.

The results in Table 4 are divided into S-SGA versus M-SGA group, S-PBIL versus M-PBIL group, S-SaDE versus M-SaDE group, and S-PSO2011 versus M-PSO2011 group. For each pair of algorithms we calculate B/S/W scores, where "B" denotes the number of times that the left algorithm performs better than the right one, "S" denotes the number of times that the left algorithm performs the same as the right one (statistically speaking), and "W" denotes the number of times that the left algorithm performs worse than the right one.

Table 4 shows that, for S-SGA versus M-SGA, the B/S/W score is 1/7/20, which indicates that S-SGA outperforms M-SGA one time, S-SGA is statistically the same as M-SGA

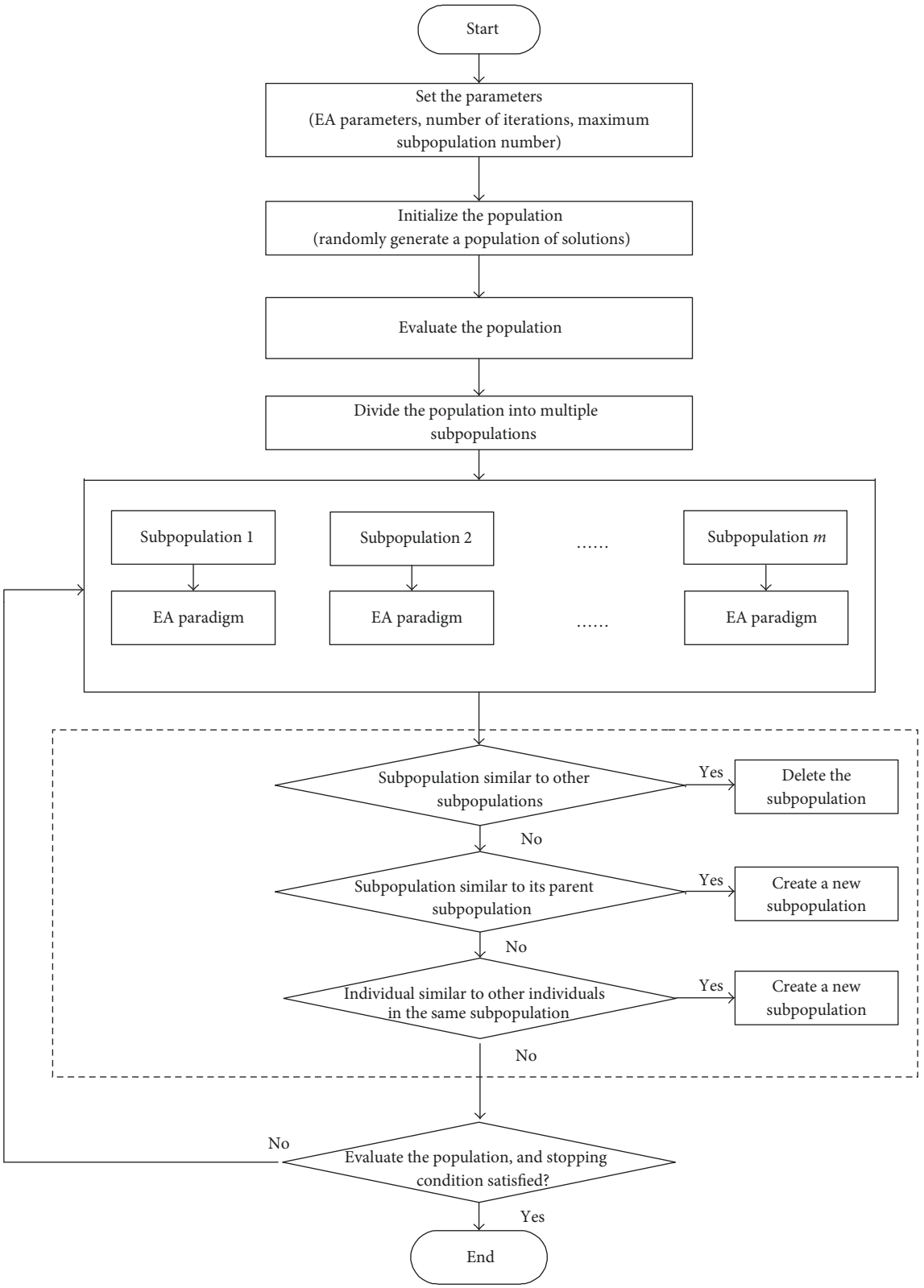

Figure 2: Flowchart of the proposed multipopulation EAs.

Table 1: 2013 CEC benchmark functions, where the search range of all functions is $-100 \leq s_i \leq 100$. More details about these functions can be found in [24].

Function		Function Name	Minimum
Unimodal functions	F1	Sphere Function	−1400
	F2	Rotated High Conditioned Elliptic Function	−1300
	F3	Rotated Bent Cigar Function	−1200
	F4	Rotated Discus Function	−1100
	F5	Different Powers Function	−1000
Basic multimodal functions	F6	Rotated Rosenbrock Function	−900
	F7	Rotated Schaffer F7 Function	−800
	F8	Rotated Ackley Function	−700
	F9	Rotated Weierstrass Function	−600
	F10	Rotated Griewank Function	−500
	F11	Rastrigin Function	−400
	F12	Rotated Rastrigin Function	−300
	F13	Discontinuous Rotated Rastrigin Function	−200
	F14	Schwefel Function	−100
	F15	Rotated Schwefel Function	100
	F16	Rotated Katsuura Function	200
	F17	Lunacek Bi_Rastrigin Function	300
	F18	Rotated Lunacek Bi_Rastrigin Function	400
	F19	Expanded Griewank plus Rosenbrock Function	500
	F20	Expanded Schaffer F6 Function	600
Composition functions	F21	Composition Function 1 ($n = 5$, rotated)	700
	F22	Composition Function 2 ($n = 3$, unrotated)	800
	F23	Composition Function 3 ($n = 3$, rotated)	900
	F24	Composition Function 4 ($n = 3$, rotated)	1000
	F25	Composition Function 5 ($n = 3$, rotated)	1100
	F26	Composition Function 6 ($n = 5$, rotated)	1200
	F27	Composition Function 7 ($n = 5$, rotated)	1300
	F28	Composition Function 8 ($n = 5$, rotated)	1400

seven times, and M-SGA outperforms S-SGA twenty times. For S-PBIL versus M-PBIL, the B/S/W score is 2/3/23, which indicates that S-PBIL outperforms M-PBIL two times, S-PBIL is statistically the same as M-PBIL three times, and M-PBIL outperforms S-PBIL twenty-three times. For S-SaDE versus M-SaDE, the B/S/W score is 0/5/23, which indicates that S-SaDE does not outperform M-SaDE in any time, S-SaDE is statistically the same as M-SaDE five times, and M-SaDE outperforms S-SaDE twenty-three times. For S-PSO2011 versus M-PSO2011, the B/S/W score is 1/5/22, which indicates that S-PSO2011 outperforms M-PSO2011 one time, S-PSO2011 is statistically the same as M-PSO2011 five times, and M-PSO2011 outperforms S-PSO2011 twenty-two times. From the results we see that multipopulation versions of these algorithms are significantly better than their single-population versions on the CEC 2013 benchmark functions, which further verifies the conclusions obtained in Tables 2 and 3. Such statistical results show that the proposed multipopulation management strategy is a good method to improve the optimization performance for EAs. The reason for its competitive performance is that multipopulation EAs effectively manage the number of subpopulations in different evolution phases throughout the evolution, which can significantly maintain population diversity, compared to single-population EAs.

4.3. Application to Complex Warehouse Scheduling Problems. In this section, the proposed multipopulation EAs are applied to a real-world complex automated warehouse scheduling problems described in [34], which are formulated as a constrained single-objective optimization problem. Warehouse scheduling in the supply chain is challenging because it is proved as a typical NP-hard problem, which is one of the most challenging types of combinatorial optimization problem [35, 36]. Layout of the warehouse system we study is shown in Figure 3, and more details about cost function, constraints, and parameter settings of the warehouse scheduling problem can be referred to [34].

In this experiment, we use the same simple-population and multipopulation EAs as those in the benchmark simulations. The constraint-handling method is based on feasibility rules by Deb [37], which has demonstrated promising performance in dealing with constraints. We consider five

Table 2: Comparisons of the best error values of the 2013 CEC benchmark functions for SGA and PBIL. In the table, "S-" and "M-" denote the single-population version and multipopulation version of algorithms, respectively. Here $[a \pm b]$ indicates the mean value and the corresponding standard deviation of 25 independent simulations. Average CPU times (minutes) are shown in the last row of the table.

Function	S-SGA	M-SGA	S-PBIL	M-PBIL
F1	$1.35E+02 \pm 5.29E+01$	$8.43E+00 \pm 1.90E+00$	$6.70E-14 \pm 4.39E-15$	$6.76E+00 \pm 7.85E+00$
F2	$3.27E+08 \pm 1.52E+07$	$5.76E+06 \pm 3.74E+05$	$5.72E+06 \pm 1.54E+05$	$3.34E+06 \pm 7.89E+05$
F3	$3.21E+10 \pm 6.65E+09$	$7.81E+10 \pm 5.48E+09$	$1.65E+09 \pm 2.48E+08$	$1.23E+08 \pm 5.54E+07$
F4	$7.20E+08 \pm 1.40E+07$	$2.64E+07 \pm 6.86E+06$	$2.32E+07 \pm 4.90E+06$	$5.70E+06 \pm 6.36E+05$
F5	$3.26E+04 \pm 5.75E+03$	$2.43E+04 \pm 1.47E+03$	$2.45E+05 \pm 5.76E+04$	$7.89E+02 \pm 2.65E+01$
F6	$9.04E+07 \pm 1.23E+06$	$6.34E+05 \pm 2.75E+04$	$1.22E+04 \pm 4.65E+03$	$4.32E+04 \pm 9.08E+03$
F7	$3.17E+07 \pm 2.44E+06$	$8.86E+05 \pm 4.23E+04$	$7.32E+06 \pm 6.87E+05$	$4.43E+05 \pm 1.24E+04$
F8	$5.32E+06 \pm 1.72E+05$	$9.09E+03 \pm 8.32E+02$	$5.70E+05 \pm 1.66E+04$	$5.54E+02 \pm 8.89E+01$
F9	$4.40E+07 \pm 8.16E+06$	$7.13E+04 \pm 7.09E+03$	$7.73E+04 \pm 1.45E+03$	$1.28E+04 \pm 5.43E+03$
F10	$8.90E+07 \pm 3.53E+06$	$2.35E+06 \pm 5.47E+05$	$3.43E+03 \pm 2.80E+02$	$6.54E+02 \pm 3.34E+01$
F11	$2.19E+06 \pm 2.80E+05$	$7.52E+07 \pm 1.66E+06$	$6.68E+04 \pm 1.58E+03$	$4.72E+02 \pm 7.80E+01$
F12	$5.43E+03 \pm 1.10E+02$	$6.60E+01 \pm 8.93E+00$	$7.70E+03 \pm 5.43E+02$	$1.23E+01 \pm 4.45E+00$
F13	$7.04E+04 \pm 5.76E+03$	$6.44E+02 \pm 1.68E+00$	$3.43E+03 \pm 6.65E+02$	$8.65E+02 \pm 3.34E+01$
F14	$5.44E+03 \pm 8.09E+02$	$5.21E+03 \pm 9.06E+02$	$7.98E+04 \pm 2.43E+03$	$1.82E+02 \pm 4.37E+01$
F15	$1.17E+05 \pm 3.56E+04$	$3.46E+05 \pm 5.32E+03$	$1.21E+05 \pm 5.63E+04$	$2.58E+05 \pm 6.74E+04$
F16	$4.56E+05 \pm 2.17E+04$	$8.97E+03 \pm 4.33E+02$	$3.65E+05 \pm 3.03E+04$	$8.91E+03 \pm 6.65E+02$
F17	$3.40E+04 \pm 8.90E+03$	$8.05E+02 \pm 1.12E+01$	$1.25E+04 \pm 7.70E+03$	$3.43E+02 \pm 1.54E+01$
F18	$1.14E+08 \pm 6.83E+07$	$3.54E+06 \pm 2.65E+05$	$2.17E+05 \pm 4.54E+04$	$7.78E+03 \pm 1.87E+02$
F19	$2.44E+07 \pm 3.27E+06$	$1.32E+07 \pm 9.04E+06$	$8.39E+05 \pm 2.78E+04$	$5.34E+03 \pm 6.65E+02$
F20	$1.81E+05 \pm 2.59E+04$	$1.88E+04 \pm 5.39E+03$	$2.12E+05 \pm 1.50E+04$	$4.23E+03 \pm 1.16E+02$
F21	$6.16E+05 \pm 7.05E+04$	$2.17E+03 \pm 3.54E+02$	$5.56E+05 \pm 4.43E+04$	$6.54E+03 \pm 1.56E+02$
F22	$9.12E+06 \pm 4.76E+05$	$2.90E+03 \pm 2.57E+02$	$4.42E+04 \pm 7.87E+03$	$3.50E+03 \pm 7.71E+02$
F23	$2.35E+05 \pm 1.32E+04$	$2.16E+06 \pm 4.56E+05$	$1.87E+04 \pm 1.21E+03$	$6.63E+04 \pm 2.19E+03$
F24	$6.88E+06 \pm 2.51E+05$	$3.32E+04 \pm 4.51E+03$	$2.33E+05 \pm 5.67E+04$	$4.35E+03 \pm 1.26E+02$
F25	$9.02E+06 \pm 1.43E+05$	$1.68E+04 \pm 6.39E+03$	$7.09E+05 \pm 4.80E+05$	$4.92E+03 \pm 3.58E+02$
F26	$2.53E+05 \pm 7.12E+04$	$7.77E+05 \pm 6.30E+04$	$1.45E+05 \pm 8.91E+04$	$1.32E+04 \pm 6.07E+03$
F27	$3.75E+06 \pm 4.66E+05$	$1.47E+04 \pm 2.83E+03$	$3.76E+04 \pm 4.32E+03$	$1.87E+02 \pm 4.16E+02$
F28	$1.89E+07 \pm 5.25E+06$	$2.41E+05 \pm 1.38E+04$	$5.98E+07 \pm 7.54E+06$	$3.65E+05 \pm 6.87E+04$
CPU time	117.86	96.35	217.85	176.37

simulation schemes. Scheme 1 ($g = 20, h = 20$) is described as follows.

p_1: (12, 6, 2, 50, 1), p_2: (51, 8, 1, 45, 1), p_3: (40, 3, 5, 46, 1), p_4: (37, 3, 4, 113, 1),

p_5: (14, 7, 1, 40, 1), p_6: (13, 5, 2, 50, 1), p_7: (45, 5, 5, 50, 1), p_8: (18, 7, 3, 33, 1),

p_9: (7, 3, 2, 35, 1), p_{10}: (11, 4, 3, 110, 1), p_{11}: (15, 2, 1, 34, 1), p_{12}: (36, 8, 4, 40, 1),

p_{13}: (8, 6, 5, 47, 1), p_{14}: (56, 2, 3, 34, 1), p_{15}: (42, 8, 3, 30, 1), p_{16}: (23, 4, 1, 50, 1),

p_{17}: (4, 6, 2, 50, 1), p_{18}: (13, 1, 1, 36, 1), p_{19}: (39, 6, 4, 42, 1), p_{20}: (45, 6, 5, 55, 1)

p_{21}: (50, 8, 1, 67, 2), p_{22}: (3, 1, 2, 74, 2), p_{23}: (55, 4, 3, 68, 2), p_{24}: (6, 7, 4, 85, 2),

p_{25}: (17, 4, 5, 74, 2), p_{26}: (60, 7, 3, 80, 2), p_{27}: (35, 6, 2, 67, 2), p_{28}: (15, 2, 1, 70, 2),

p_{29}: (57, 4, 2, 80, 2), p_{30}: (2, 1, 4, 62, 2), p_{31}: (25, 8, 2, 76, 2), p_{32}: (5, 2, 4, 64, 2),

p_{33}: (17, 5, 3, 76, 2), p_{34}: (41, 2, 5, 91, 2), p_{35}: (19, 3, 2, 70, 2), p_{36}: (20, 1, 1, 82, 2),

p_{37}: (42, 6, 2, 88, 2), p_{38}: (32, 6, 3, 75, 2), p_{39}: (9, 7, 1, 82, 2), p_{40}: (58, 5, 4, 69, 2)

In Scheme 1, g is the number of storage products, and h is the number of retrieval products. p_i is the storage unit, and it is denoted by (x, y, z, w, θ), where x, y, and z are the Euclidean coordinates of each storage unit, w is the weight coefficient that affects the scheduling quality, and $\theta = 1$ indicates a storage product and $\theta = 2$ indicates a retrieval product.

Scheme 2 ($g = 20, h = 16$) includes all the storage products of Scheme 1 but only the first 16 retrieval products of Scheme 1. Scheme 3 ($g = 20, h = 12$) includes all the storage products of Scheme 1 but only the first 12 retrieval products. Scheme 4 ($g = 16, h = 20$) includes the first 16 storage products of Scheme 1 and all of the retrieval products. Scheme 5 ($g = 12, h = 20$) includes the first 12 storage products of Scheme 1 and all of the retrieval products.

Optimization results of different simple-population and multipopulation EAs are summarized in Table 5. From the

Table 3: Comparisons of the best error values of the 2013 CEC benchmark functions for SaDE and PSO2011. In the table, "S-" and "M-" denote the single-population version and multipopulation version of algorithms, respectively. Here $[a \pm b]$ indicates the mean value and the corresponding standard deviation of 25 independent simulations. The best result in each row (Tables 2 and 3 combined) is shown in *bold font*. Average CPU times (minutes) are shown in the last row of the table.

Function	S-SaDE	M-SaDE	S-PSO2011	M-PSO2011
F1	$5.43E-10 \pm 1.36E-11$	**0.00E+00 ± 0.00E+00**	$6.89E-19 \pm 7.97E-20$	**0.00E+00 ± 0.00E+00**
F2	$4.25E+05 \pm 3.12E+04$	$3.92E+05 \pm 5.76E+04$	$1.32E+06 \pm 4.80E+05$	**1.34E+05 ± 5.67E+04**
F3	$5.99E+07 \pm 3.76E+06$	$1.89E+07 \pm 7.54E+06$	$7.86E+07 \pm 9.92E+06$	**2.43E+06 ± 4.72E+05**
F4	$1.25E+06 \pm 4.68E+05$	**1.40E+05 ± 4.35E+04**	$1.17E+06 \pm 5.54E+05$	$6.76E+06 \pm 5.31E+05$
F5	$5.34E+02 \pm 3.70E+01$	$7.44E+00 \pm 2.31E+00$	$1.15E+02 \pm 6.18E+01$	**0.00E+00 ± 0.00E+00**
F6	$1.45E+02 \pm 6.34E+01$	**5.67E+00 ± 4.78E+00**	$5.37E+04 \pm 6.76E+03$	$4.32E+03 \pm 7.98E+02$
F7	$2.91E+02 \pm 4.56E+01$	**7.07E+00 ± 8.92E−01**	$1.25E+04 \pm 7.08E+03$	$1.54E+02 \pm 6.97E+01$
F8	$8.32E+03 \pm 1.57E+02$	$5.65E+01 \pm 1.23E+00$	$9.43E+02 \pm 2.12E+01$	**2.82E+01 ± 9.33E+00**
F9	$6.43E+03 \pm 5.54E+02$	**3.47E+01 ± 6.38E+00**	$4.41E+03 \pm 3.56E+02$	$5.23E+02 \pm 4.34E+01$
F10	$2.32E+03 \pm 1.17E+02$	**7.81E+00 ± 1.22E−01**	$3.76E+01 \pm 1.89E+00$	$5.67E+01 \pm 3.21E+00$
F11	$6.33E+02 \pm 2.25E+01$	**8.90E+00 ± 3.86E+00**	$6.65E+03 \pm 8.54E+02$	$1.45E+01 \pm 2.54E+00$
F12	$4.19E+03 \pm 4.32E+02$	**7.78E+01 ± 5.47E+00**	$2.43E+03 \pm 1.26E+02$	$7.87E+01 \pm 6.23E+01$
F13	$5.76E+03 \pm 2.45E+02$	**6.60E+01 ± 3.71E+00**	$6.54E+04 \pm 6.13E+03$	$1.34E+02 \pm 8.98E+01$
F14	$2.65E+01 \pm 1.36E+00$	**5.65E+01 ± 8.89E+00**	$5.48E+02 \pm 3.87E+01$	$6.34E+01 \pm 5.26E+00$
F15	$5.20E+04 \pm 6.65E+03$	**4.36E+03 ± 2.17E+02**	$4.19E+05 \pm 6.22E+04$	$4.55E+04 \pm 3.29E+03$
F16	$3.76E+03 \pm 6.72E+02$	**5.54E+00 ± 2.16E+00**	$1.00E+03 \pm 4.55E+02$	$7.40E+02 \pm 1.23E+01$
F17	$4.32E+03 \pm 9.24E+02$	$4.45E+01 \pm 3.92E-01$	$1.32E+02 \pm 7.64E+01$	**6.76E+00 ± 1.65E−01**
F18	$1.18E+03 \pm 4.80E+02$	**1.23E+01 ± 7.89E+00**	$2.28E+03 \pm 6.59E+02$	$2.45E+02 \pm 7.89E+01$
F19	$5.92E+02 \pm 6.43E+01$	**7.76E+01 ± 4.37E+00**	$8.98E+03 \pm 1.66E+02$	$5.36E+02 \pm 4.75E+01$
F20	$1.32E+01 \pm 5.44E+00$	**6.67E+01 ± 3.41E+00**	$1.45E+03 \pm 9.81E+02$	$2.38E+02 \pm 7.62E+01$
F21	$3.46E+03 \pm 1.84E+02$	**2.48E+02 ± 7.80E+01**	$5.99E+04 \pm 4.13E+03$	$5.43E+02 \pm 8.90E+01$
F22	$7.79E+02 \pm 5.62E+01$	**1.22E+01 ± 3.17E+00**	$1.76E+03 \pm 3.32E+02$	$5.32E+03 \pm 4.54E+02$
F23	$2.33E+05 \pm 1.75E+04$	$8.92E+03 \pm 3.44E+03$	$5.53E+05 \pm 6.96E+04$	**1.26E+03 ± 3.33E+02**
F24	$4.46E+04 \pm 8.08E+03$	$7.28E+02 \pm 5.38E+00$	$6.87E+06 \pm 5.43E+05$	**6.43E+02 ± 9.09E+01**
F25	$3.42E+04 \pm 1.37E+03$	$9.03E+02 \pm 1.28E+01$	$9.06E+05 \pm 8.85E+04$	**1.21E+02 ± 4.58E+01**
F26	$6.60E+02 \pm 2.76E+01$	**4.22E+02 ± 5.76E+01**	$1.79E+03 \pm 5.41E+02$	$2.14E+03 \pm 8.66E+02$
F27	$2.52E+03 \pm 4.41E+02$	$8.09E+02 \pm 8.90E+01$	$3.99E+04 \pm 3.72E+03$	**6.43E+02 ± 1.37E+01**
F28	$7.72E+03 \pm 8.58E+02$	**3.60E+02 ± 5.81E+01**	$1.18E+04 \pm 2.56E+03$	$9.13E+03 \pm 6.55E+02$
CPU time	185.32	149.05	138.70	118.52

table, it is seen that M-SaDE performs best for all of scheme cases because of its lowest scheduling quality effect. Furthermore, we find that multipopulation EAs perform significantly better than single-population EAs on these scheme cases of the warehouse scheduling problem, which again verifies the conclusions obtained in the test of benchmark functions.

A sample multipopulation SaDE scheduling route output is shown in Table 6 for Scheme 1. It is seen that the warehouse scheduling problem is divided into 5 routes, where machine 1 implements routes 1 and 2, and machine 2 implements routes 3, 4, and 5. Each route includes 8 storage units, where the first 4 storage units are used to store products, and the last 4 storage units are used to retrieve products.

5. Conclusions

In this paper, we first propose a management strategy for the number of multiple subpopulations based on individual distance and population distance, and it dynamically increases or decreases the subpopulation number during evolution process to maintain population diversity. Then we integrate the proposed multipopulation management strategy into EAs, including SGA, PBIL, SaDE, and PSO2011, to develop new multipopulation EAs. Next, the proposed multipopulation EAs are tested on CEC benchmark functions, and empirical results show that any single-population EA can be easily extended to a multipopulation EA by the proposed strategy, and the proposed multipopulation methods can obtain the better optimization performance than single-population EAs. Finally, these multipopulation EAs are used to solve real-world complex automated warehouse scheduling problems, and experimental results show that the proposed multipopulation EAs can obtain satisfied solutions.

In future research, at least three directions are envisioned. First, the proposed multipopulation management strategy has been combined into several EAs and improves their optimization performance. The multipopulation management strategy presented here could be extended for more EAs

Table 4: Wilcoxon test results for pairwise algorithm comparisons. If the difference between the algorithms is statistically significant, the pairs are marked as follows: "x-o" shows that the left algorithm is better than the right one; "o-x" shows that the right algorithm is better than the left one. The B/S/W row at the bottom shows the total scores, where "B" denotes the number of times the left algorithm performs better, "S" denotes the number of times the two algorithms perform the same, and "W" denotes the number of times the left algorithm performs worse than the right one.

Function	S-SGA versus M-SGA	S-PBIL versus M-PBIL	S-SaDE versus M-SaDE	S-PSO2011 versus M-PSO2011
F1	o-x	o-x	o-x	o-x
F2	o-x	–	–	–
F3	–	o-x	–	o-x
F4	o-x	o-x	o-x	–
F5	–	o-x	o-x	o-x
F6	o-x	x-o	o-x	o-x
F7	o-x	o-x	o-x	o-x
F8	o-x	o-x	o-x	o-x
F9	o-x	–	o-x	o-x
F10	o-x	o-x	o-x	–
F11	x-o	o-x	o-x	o-x
F12	o-x	o-x	o-x	o-x
F13	o-x	o-x	o-x	o-x
F14	–	o-x	–	o-x
F15	–	–	o-x	o-x
F16	o-x	o-x	o-x	–
F17	o-x	o-x	o-x	o-x
F18	o-x	o-x	o-x	o-x
F19	–	o-x	o-x	o-x
F20	o-x	o-x	–	o-x
F21	o-x	o-x	o-x	o-x
F22	o-x	o-x	o-x	x-o
F23	–	x-o	o-x	o-x
F24	o-x	o-x	o-x	o-x
F25	o-x	o-x	o-x	o-x
F26	–	o-x	–	–
F27	o-x	o-x	o-x	o-x
F28	o-x	o-x	o-x	o-x
B/S/W	1/7/20	2/3/23	0/5/23	1/5/22

Table 5: Optimization results for 5 schemes of the warehouse scheduling problem. Here $a(b)$ denotes the mean value and corresponding standard deviation of the scheduling quality effect. The best results in each row are shown in *boldface* font.

Problem	(g, h)	S-SGA	M-SGA	S-PBIL	M-PBIL
Scheme 1	(20, 20)	2629.3 (75.4)	2317.7 (47.6)	2613.1 (55.9)	2304.1 (28.3)
Scheme 2	(20, 16)	1627.1 (66.2)	1423.7 (48.6)	1599.4 (57.3)	1398.4 (66.4)
Scheme 3	(20, 12)	914.5 (32.7)	852.3 (53.2)	973.4 (44.1)	886.0 (53.7)
Scheme 4	(16, 20)	1533.2 (82.1)	1295.4 (59.6)	1436.1 (72.4)	1266.7 (36.7)
Scheme 5	(12, 20)	884.4 (26.4)	712.6 (52.0)	804.3 (36.7)	699.4 (45.1)
Problem	(g, h)	S-SaDE	M-SaDE	S-PSO2011	M-PSO2011
Scheme 1	(20, 20)	2322.5 (79.3)	**2003.7** (56.6)	2395.6 (47.5)	2113.8 (39.1)
Scheme 2	(20, 16)	1453.6 (46.2)	**1205.4** (47.3)	1490.1 (29.3)	1347.6 (61.7)
Scheme 3	(20, 12)	812.6 (39.8)	**705.3** (55.4)	884.2 (16.7)	783.3 (24.5)
Scheme 4	(16, 20)	1233.8 (38.0)	**1100.9** (41.2)	1312.5 (53.4)	1194.2 (29.9)
Scheme 5	(12, 20)	711.5 (36.7)	**621.8** (55.1)	746.3 (28.4)	686.3 (33.1)

Table 6: Scheduling orders as optimized by multipopulation SaDE. "Machine" denotes the machine number index, "Route" denotes the route number index, and "Scheduling orders" denote the scheduling orders that one machine implements in one route.

Machine index	Route	Scheduling orders
Machine 1	1	$p_{11} \to p_2 \to p_9 \to p_{15} \to p_{34} \to p_{29} \to p_{22} \to p_{37}$
	2	$p_5 \to p_8 \to p_{19} \to p_{13} \to p_{27} \to p_{35} \to p_{28} \to p_{32}$
Machine 2	3	$p_1 \to p_{16} \to p_4 \to p_{12} \to p_{24} \to p_{26} \to p_{36} \to p_{38}$
	4	$p_{17} \to p_6 \to p_{10} \to p_{18} \to p_{31} \to p_{23} \to p_{40} \to p_{25}$
	5	$p_7 \to p_{14} \to p_{20} \to p_3 \to p_{30} \to p_{21} \to p_{39} \to p_{33}$

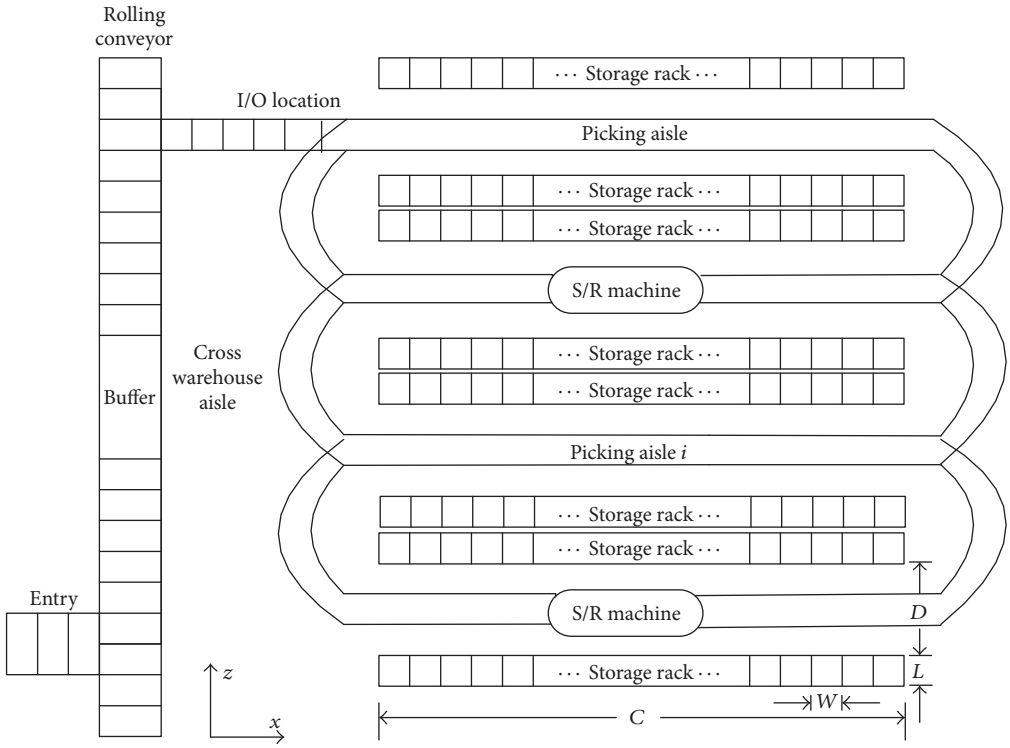

Figure 3: Layout of the warehouse system.

and swarm intelligence algorithms, for example, fireworks algorithm (FWA) [38, 39], brain storm optimization (BSO) algorithm [40, 41], teaching-learning based optimization (TLBO) algorithm [42–44], and Jaya optimization algorithm [45]. Second, in this paper, we do not consider the communication between subpopulations. Undoubtedly, the interaction of subpopulations is important to enhance optimization performance by information share between subpopulations. So how to implement the adaptive interaction of subpopulations is additional direction for future study. Third, solving real-world application problems is the perpetual goal for EAs, and it would be fruitful to apply the proposed multipopulation EAs to various complex real-world problems. There is no doubt that more applications of the proposed multipopulation EAs can emerge in the near future with focused research.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] H. Ma and D. Simon, *Evolutionary Computation with Biogeography-based Optimization*, John Wiley & Sons, New Jersey, NJ, USA, 2017.

[2] Z. Yang, K. Li, Q. Niu, and Y. Xue, "A comprehensive study of economic unit commitment of power systems integrating various renewable generations and plug-in electric vehicles," *Energy Conversion and Management*, vol. 132, pp. 460–481, 2017.

[3] H. Ma, S. Ye, D. Simon, and M. Fei, "Conceptual and numerical comparisons of swarm intelligence optimization algorithms," *Soft Computing*, vol. 21, no. 11, pp. 3081–3100, 2017.

[4] X. Li, J. Lai, and R. Tang, "A hybrid constraints handling strategy for multiconstrained multiobjective optimization problem of Microgrid economical/environmental dispatch," *Complexity*, Article ID 6249432, 12 pages, 2017.

[5] Z. Yang, K. Li, Q. Niu, and Y. Xue, "A novel parallel-series hybrid meta-heuristic method for solving a hybrid unit commitment problem," *Knowledge-Based Systems*, vol. 134, pp. 13–30, 2017.

[6] P. Siarry, A. Pétrowski, and M. Bessaou, "A multipopulation genetic algorithm aimed at multimodal optimization," *Advances in Engineering Software*, vol. 33, no. 4, pp. 207–213, 2002.

[7] J. Alami, A. E. Imrani, and A. Bouroumi, "A multipopulation cultural algorithm using fuzzy clustering," *Applied Soft Computing*, vol. 7, no. 2, pp. 506–519, 2007.

[8] W.-D. Chang, "A modified particle swarm optimization with multiple subpopulations for multimodal function optimization problems," *Applied Soft Computing*, vol. 33, pp. 170–182, 2015.

[9] S. K. Nseef, S. Abdullah, A. Turky, and G. Kendall, "An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems," *Knowledge-Based Systems*, vol. 104, pp. 14–23, 2016.

[10] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Information Sciences*, vol. 329, pp. 329–345, 2016.

[11] H. Pourvaziri and B. Naderi, "A hybrid multi-population genetic algorithm for the dynamic facility layout problem," *Applied Soft Computing*, vol. 24, pp. 457–469, 2014.

[12] B. Niu, H. Huang, L. Tan, and Q. Duan, "Symbiosis-Based Alternative Learning Multi-Swarm Particle Swarm Optimization," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 1, pp. 4–14, 2017.

[13] J. Xiao, W. Li, B. Liu, and P. Ni, "A novel multi-population coevolution immune optimization algorithm," *Soft Computing*, 2015.

[14] A. M. Turky and S. Abdullah, "A multi-population harmony search algorithm with external archive for dynamic optimization problems," *Information Sciences*, vol. 272, pp. 84–95, 2014.

[15] N. S. Jaddi, S. Abdullah, and A. Hamdan, "Multi-population cooperative bat algorithm-based optimization of artificial neural network model," *Information Sciences*, vol. 294, pp. 628–644, 2015.

[16] F. B. Ozsoydan and A. Baykasoğlu, "A multi-population firefly algorithm for dynamic optimization problems," in *Proceedings of the IEEE International Conference on Evolving and Adaptive Intelligent Systems, EAIS 2015*, France, December 2015.

[17] G. Mauša and T. Galinac Grbac, "Co-evolutionary multi-population genetic programming for classification in software defect prediction: An empirical case study," *Applied Soft Computing*, vol. 55, pp. 331–351, 2017.

[18] C. Li, T. T. Nguyen, M. Yang, S. Yang, and S. Zeng, "Multi-population methods in unconstrained continuous dynamic environments: The challenges," *Information Sciences*, vol. 296, no. 1, pp. 95–118, 2015.

[19] R. Shang, Y. Wang, J. Wang, L. Jiao, S. Wang, and L. Qi, "A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem," *Information Sciences*, vol. 277, pp. 609–642, 2014.

[20] H. Cheng, S. Yang, and X. Wang, "Immigrants-enhanced multi-population genetic algorithms for dynamic shortest path routing problems in mobile ad hoc networks," *Applied Artificial Intelligence*, vol. 26, no. 7, pp. 673–695, 2012.

[21] C. Li and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," in *Proceedings of the 11th IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 439–446, IEEE Press, Piscataway, NJ, USA, May 2009.

[22] X. P. Wang and L. X. Tang, "An adaptive multi-population differential evolution algorithm for continuous multi-objective optimization," *Information Sciences*, vol. 348, pp. 124–141, 2016.

[23] H. Kwasnicka and M. Gierusz, "Managing of cooperative genetic algorithms by intelligent agent," in *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, ISDA '05*, pp. 564–569, Poland, September 2005.

[24] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernandez-Diaz, "Problem definitions and evaluation criteria for the CEC 2013 special session and competition on real-parameter optimization," Technical Report, Nanyang Technological University, Singapore, pp. 1–39, 2013.

[25] W. Khatib and P. J. Fleming, "The Stud GA: A mini revolution?" in *Parallel Problem Solving from Nature*, pp. 683–691, Springer, New York, NY, USA, 1998.

[26] L. Pedro and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, Boston, MA, USA, 2002.

[27] S.-Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large-scale optimization," *Soft Computing*, vol. 15, no. 11, pp. 2175–2185, 2011.

[28] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[29] M. Omran and M. Clerc, Particle Swarm Central. http://www.particleswarm.info/, 2011.

[30] K. Yu, X. Wang, and Z. Wang, "Multiple learning particle swarm optimization with space transformation perturbation and its application in ethylene cracking furnace optimization," *Knowledge-Based Systems*, 2015.

[31] M. M. Al-Rifaie and T. Blackwell, "Bare bones particle swarms with jumps," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7461, pp. 49–60, 2012.

[32] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[33] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[34] H. Ma, S. Su, D. Simon, and M. Fei, "Ensemble multi-objective biogeography-based optimization with application to automated warehouse scheduling," *Engineering Applications of Artificial Intelligence*, vol. 44, article no. 2340, pp. 79–90, 2015.

[35] J.-P. Gagliardi, J. Renaud, and A. Ruiz, "Models for automated storage and retrieval systems: A literature review," *International Journal of Production Research*, vol. 50, no. 24, pp. 7110–7125, 2012.

[36] A. T. Almaktoom, "Stochastic reliability measurement and design optimization of an inventory management system," *Complexity*, Article ID 1460163, Art. ID 1460163, 9 pages, 2017.

[37] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[38] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 6145, no. 1, pp. 355–364, 2010.

[39] S. Ye, H. Ma, S. Xu, W. Yang, and M. Fei, "An effective fireworks algorithm for warehouse-scheduling problem," *Transactions of the Institute of Measurement and Control*, vol. 39, no. 1, pp. 75–85, 2017.

[40] S. Cheng, Q. Qin, J. Chen, and Y. Shi, "Brain storm optimization algorithm: a review," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 445–458, 2016.

[41] Y. H. Shi, "Brain storm optimization algorithm," in *Proceedings of the International Conference in Swarm Intelligence*, vol. 6728, pp. 303–309, Springer, Heidelberg, Germany, 2011.

[42] Z. Yang, K. Li, Q. Niu, Y. Xue, and A. Foley, "A self-learning teaching-learning based optimization for dynamic economic/environmental dispatch considering multiple plug-in electric vehicle loads," *Journal of Modern Power System and Clean Energy*, vol. 2, no. 4, pp. 298–307, 2014.

[43] K. Yu, X. Wang, and Z. Wang, "An improved teaching-learning-based optimization algorithm for numerical and engineering optimization problems," *Journal of Intelligent Manufacturing*, vol. 27, no. 4, pp. 831–843, 2016.

[44] K. Yu, X. Chen, X. Wang, and Z. Wang, "Parameters identification of photovoltaic models using self-adaptive teaching-learning-based optimization," *Energy Conversion and Management*, vol. 145, pp. 233–246, 2017.

[45] K. Yu, J. Liang, B. Qu, X. Chen, and H. Wang, "Parameters identification of photovoltaic models using an improved JAYA optimization algorithm," *Energy Conversion and Management*, vol. 150, pp. 742–753, 2017.