A Multi-population Based Multi-objective Evolutionary Algorithm

Haiping Ma, Minrui Fei, Zheheng Jiang, Ling Li, Huiyu Zhou, Danny Crookes, *Senior Member*, *IEEE*

***Abstract*—Multi-population is an effective optimization component often embedded into evolutionary algorithms to solve optimization problems. In this paper, a new multi-population based multi-objective genetic algorithm is proposed, which uses a unique cross-subpopulation migration process inspired by biological processes to share information between subpopulations. Then, a Markov model of the proposed multi-population multi-objective genetic algorithm is derived, the first of its kind, which provides an exact mathematical model for each possible population occurring simultaneously with multiple objectives. Simulation results of two multi-objective test problems with multiple subpopulations justify the derived Markov model, and show that the proposed multi-population method can improve the optimization ability of the multi-objective genetic algorithm. Also, the proposed multi-population method is applied to other multi-objective evolutionary algorithms for evaluating its performance against the IEEE Congress on Evolutionary Computation multi-objective benchmarks. The experimental results show that a single-population multi-objective evolutionary algorithm can be extended to a multi-population version, while obtaining better optimization performance.**

***Index Terms*—Evolutionary algorithm, Multi-population, Multi-objective optimization, Genetic algorithms, Markov chain**

I. Introduction

Many real-world optimization problems require us to deal with multiple objectives simultaneously, where it is not easy to reach agreement. These problems are referred to as multi-objective optimization problems (MOPs) in the scientific domain [1-3], and their non-dominated solutions are regarded as Pareto-optimal solutions. Without loss of generality, an MOP is a minimization problem for each objective, which is formulated as follows [4, 5]:

$$\begin{aligned} &\text{minimize} \quad F(x) = \left(f_1(x), f_2(x), \cdots, f_m(x)\right)^T \\ &\text{subject to} \quad x \in \Omega \end{aligned} \tag{1}$$

That is, an MOP is to simultaneously minimize all m functions $F(x)$. In Equation (1), $\Omega = \{x \in R^n\}$ is the search space and x is the decision variable vector. $F: \Omega \to R^m$ maps the n-dimensional decision space Ω to the m-dimensional objective space R^m. Because of the complexity of MOPs in real-world applications and limited computational resources, true Pareto-optimal solutions are difficult to obtain analytically or exactly. Instead, multi-objective evolutionary algorithms (MOEAs) have been developed and have proved promising for solving these problems [6, 7]. In [8], Deb proposed a fast and elite non-dominated sorting genetic algorithm II (NSGA-II), which is one of the representative state-of-the-art multi-objective evolutionary algorithms. In [9], Zhang presented a well accepted multi-objective evolutionary algorithm based on decomposition (MOEA/D), which was different from NSGA-II, being based on conventional aggregation approaches where an MOP was decomposed into a number of scalar objective optimization problems.

Meantime, many multi-population methods inspired by biological or natural evolution processes have been embedded into evolutionary algorithms to solve various simple-objective problems (SOPs), and to obtain satisfactory optimization performance [10, 11]. Moreover, recently, multi-population methods have demonstrated promising results for solving MOPs [12, 13], while achieving satisfactory trade-offs between meeting the different objectives. In solving optimization problems, multi-population methods firstly decompose the initial population into several small subpopulations. Then some evolving operators, for example, recombination and mutation for genetic algorithms (GAs), are executed to implement the evolution. Finally, different subpopulations work together in order to search in different local areas to render Pareto-optimal solutions.

In some publications, researchers have developed MOEAs with multiple subpopulations, and the experimental results show that the performance of MOEAs can be considerably improved. In [14], the authors proposed a multi-population genetic algorithm (MPGA) to solve multi-objective scheduling problems for parallel computers, where each subpopulation evolved separately while an elite strategy was used to select the best solutions for each objective and the best solution of the combined objective. Simulation results showed MPGA had better performance over a wide range of problems, compared with the regular multi-objective genetic algorithms (MOGAs). Another adaptive MPGA was proposed in [15] to solve the multi-objective group scheduling problem in hybrid and flexible flowshops with sequence-dependent setups, where subpopulations were generated by the re-arrangement of initial solutions, and computational results showed that the proposed algorithm performed better than the other algorithms. In [16], the authors used multi-population multi-objective EAs for optimizing NC-tool paths for simultaneous five-axis milling, where the topology with subpopulations was considered within the problem-specific restriction area. Experiments on different

Manuscript received ***. This material was supported in part by the National Natural Science Foundation of China under Grant Nos. 61640316 and 61633016, and the Fund for China Scholarship Council under Grant No. 201608330109. H. Zhou is supported in part by UK EPSRC under Grant EP/N011074/1 and Royal Society-Newton Advanced Fellowship under Grant NA160342.

Haiping Ma is with the Department of Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang, 312000, China. (e-mail: Mahp@usx.edu.cn).

Minrui Fei is with Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, 200072, China. (e-mail: mrfei@staff.shu.edu.cn).

Zheheng Jiang and Huiyu Zhou are with Department of Informatics, University of Leicester, LE1 7RH, UK (e-mail: {zj53, hz143}@leicester.ac.uk).

Ling Li is with the School of Computing, University of Kent, ME4 4AG, UK (C.Li@kent.ac.uk)

Danny Crookes is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT3 9DT, UK (e-mail: d.crookes@qub.ac.uk).

NC-paths structures showed that the proposed method obtained promising results. In [17], a multi-population cooperative co-evolutionary algorithm (MPCCA) was proposed for the multi-objective capacitated arc routing problem. In MPCCA, the population was divided into multiple subpopulations using the divide-and-conquer method based on their different direction vectors. Each subpopulation evolved separately in each generation and the adjacent subpopulations could share their individuals in the form of cooperative subpopulations. Experimental results show that the proposed method obtained better performance than the other algorithms. A multi-objective multi-population biased random-key genetic algorithm was presented in [18] for the 3D container loading problem, where several subpopulations were evolved independently in parallel, and all the solutions in each subpopulation were ranked by fitness. Then the overall top solutions were added into each subpopulation. Numerical experiments showed the viability of the proposed method. An adaptive multi-population differential evolution (AMPDE) algorithm was proposed in [19] for continuous multi-objective optimization, where the size of each subpopulation was adaptively adjusted based on the information derived from the search data. Computational results showed that the proposed AMPDE was superior to the previous MOEAs.

Although MOEAs combined with multi-population methods have demonstrated promising outcomes, most of them have the following unsolved issues. First, most multi-population MOEAs do not have enough autonomous and efficient communication between subpopulations. That is, the communication is only to exchange individuals based on some simple connection topologies (defined as the connection between the subpopulations) with a certain communication interval (which is a parameter that controls how often the communication occurs) and communication rate (which is a parameter that controls how many individuals communicate). We believe that the communication between subpopulations can be used to increase population diversity for MOPs [20, 21], and hence, this will accelerate the optimality search to find better solutions. However, it is very difficult to make a judicious choice for the communication parameters, before MOPs can be solved.

Second, previous studies mainly focus on developing a specific multi-population MOEA rather than looking at the bigger picture of how multi-population methods would affect the MOEA performance and how they can be addressed in a generic way. Although some methods, e.g. [22-24], provide guidelines to empirically choose multi-population parameters for SOPs, little empirical analysis has been done to investigate the impact of multi-population for MOPs. Further, for either multi-population SOPs or multi-population MOPs, there has been no theoretical model to analyze the effect of multi-population, and to understand intrinsic differences and similarities between multi-population and traditional MOEAs.

In order to address these challenging issues, this paper, the first of its kind, provides comprehensive discussion of these problems and corresponding solutions, supported by both mathematical models and empirical results. This paper also presents constructive comments and suggestions on future MOEA designs that may contribute to the final solution of the challenges.

Our work is different from previous studies in MOEAs in the following aspects. First, we introduce a new multi-population MOGA, in which a new operator called cross-subpopulation migration is used to communicate between subpopulations to exchange information. Here migration uses the rank of solution fitness, objective similarity and Euclidian distances between subpopulations to adaptively determine the system parameters, including connection topology, communication frequency and communication rate. The advantage of this operation is that the system can be more autonomous and more efficient.

Second, a mathematical model for the proposed multi-population MOGA is derived to reveal the algorithmic characteristics and justify the algorithm performance. It is important to design an algorithm to find better tuning parameters. More generally, the model of multi-population MOGA is useful in producing insights into how the algorithm behaves and when it is likely to make an impact.

The novel contributions of this paper include the following aspects: (a) it presents a new multi-population method used in MOGA, in which cross-subpopulation migration is used to maintain population diversity; (b) it derives the Markov model for the proposed multi-population MOGA, which for the first time constructs a mathematical model of the multi-population method; (c) it uses simulations to verify the derived model, and theoretically explores the effect of the multi-population method on algorithmic performance; (d) it investigates the optimization ability of the proposed multi-population MOGA for solving a set of multi-objective benchmarks; and (e) although in this paper the proposed multi-population method is only used in MOGA at this stage, our idea can be extended to any MOEA without changing the original structure of other algorithms.

The remainder of the paper is organized as follows. Section II gives detailed descriptions of the proposed multi-population MOGA method. Section III presents the Markov model of the proposed multi-population MOGA approach. Section IV presents simulations to verify the proposed model, and show the optimization performance of the proposed algorithm on multi-objective benchmarks. Section V gives conclusions and suggestions for future research.

II. MULTI-POPULATION MOGA

This section firstly presents the foundation of the proposed multi-population MOGA (Section *A*), in which cross-subpopulation migration, used to naturally communicate among subpopulations, is introduced. Then it presents the implementation of the proposed multi-population MOGA in detail (Section *B*).

A. Foundation of Multi-population MOGA

Like most heuristic algorithms, multi-population MOGA is inspired by nature. The environment of multi-population MOGA is analogous to island models used in other evolutionary computations, which are effective tools for parallel computing. Each island is considered as a possible solution to the problem, and a group of islands represent a subpopulation.

In a multi-population MOGA method, it firstly performs selection and crossover for each subpopulation, equivalent to regular GAs, which is called 'within-subpopulation genetic operator'. Then candidate solutions for subpopulations are exchanged by migration, which is an operator inspired by species migration between islands [25, 26], called cross-subpopulation migration because of the migration occurring between subpopulations. The purpose of the

migration is to facilitate information sharing, and consequently it preserves diversity throughout the entire population. So the cross-subpopulation migration is an important operator in the proposed multi-population MOGA. Finally, it performs mutation, which is the same as those used in regular GAs. The main difference between multi-population and single-population MOGAs is that the former uses adaptive interactions between subpopulations to enhance the exploration process. A framework for multi-population MOGA is illustrated in Figure 1.

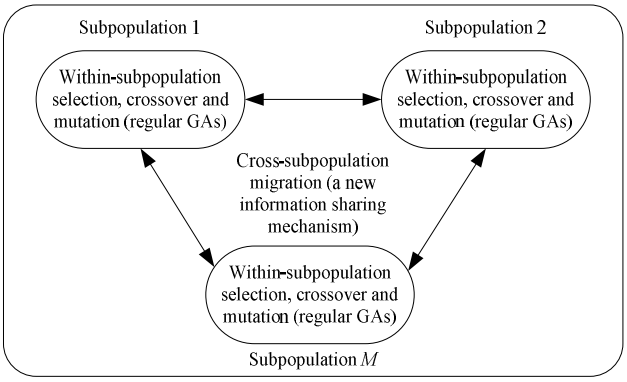

Fig.1. A framework of a multi-population MOGA including the within-subpopulation genetic operator (selection, crossover and mutation), and cross-subpopulation migration.

B. *Implementation of multi-population MOGA*

Within-subpopulation genetic operator mainly includes selection and crossover. Within-subpopulation *selection* is implemented by roulette-wheel selection [27], and it uses a probability distribution, which is linearly related to the solution ranks. In MOGA, solution ranking often uses classical non-dominated sorting [8], which reflects the relative performance of each solution in a population, an effective method in the multi-objective optimization domain. After the parent solutions are selected for recombination, we perform within-subpopulation *crossover*. Here we use global uniform crossover, which means that many parents can contribute solution variables to a single offspring, and each solution variable in an offspring is generated independently from every other solution variable.

Cross-subpopulation migration is implemented by probabilistically choosing the replaced solution based on the solution ranks. Then we find pairs of subpopulations that are suitable for migration based on the objective similarity levels of subpopulations. That is, we use the objective similarity levels to decide which subpopulations to migrate to or from. The reason is that subpopulations with a high objective similarity are more likely to benefit each other through migration than the subpopulations with a low similarity. The similarity levels are calculated using Algorithm 1, where G and H are the sets of objective costs of two solutions in different subpopulations. The pair probability P_{sub} between the subpopulations is proportional to the maximum objective similarity level in the population, which is calculated as follows:

$$P_{sub} = \frac{SL}{SL_{\max}} \tag{2}$$

where SL is the objective similarity level between two solutions cross subpopulations, and SL_{max} is the maximum objective similarity level in the population.

Algorithm 1 – Similarity level calculation cross subpopulations

```
Set the objective similarity level SL = 0;
For each g ∈ G, where G is the objective set of one solution
  For each h ∈ H, where H is the objective set of another solution
    If g and h are the same then
      SL = SL + 1;
    End if
  End for
End for
```

Once we obtain the pair of subpopulations to migrate to and from, we calculate the Euclidian distance between the replaced solution and each solution in the selected subpopulation. The introduction of Euclidian distance is based on the concept of diversity: a larger diversity in a population provides more opportunities to find an optimal solution [20]. We then use roulette-wheel selection based on Euclidian distance to select the emigrating solutions in the selected subpopulation. Figure 2 shows an example of a selected emigrating solution replacing a solution in the immigrating subpopulation, where we firstly calculate the Euclidian distance between the replaced solution in the immigrating subpopulation and each solution in the emigrating subpopulation. Then we create the roulette-wheel probability D based on the Euclidian distance, and select the emigrating solution from the emigrating subpopulation based on roulette-wheel selection.

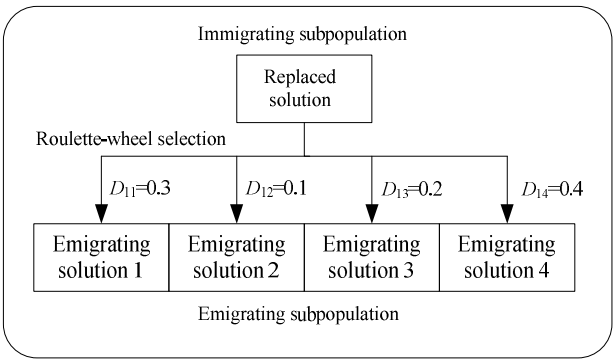

Fig.2. An example of emigrating solution selection across an immigrating subpopulation and an emigrating subpopulation.

Finally, we perform migration between the replaced solution y_k and the emigrating solution y_j cross the subpopulations. Migration is denoted as

$$y_k(s) \leftarrow y_j(s) \tag{3}$$

where s is a solution feature index. Equation (3) states that a solution variable in the emigrating solution replaces one in the replaced solution. In cross-subpopulation migration, each solution variable in each replaced solution in a subpopulation has a chance to be replaced by a solution variable of an emigrating solution from another subpopulation.

Mutation in multi-population MOGA is identical to that of regular GAs, which randomly modifies a solution variable based on the mutation probability. The purpose of mutation is also to increase population diversity [27].

The description of one generation of multi-population MOGA is shown in Algorithm 2, where y is the entire population of all the subpopulations, z is the temporary population corresponding to y, y_{ik} is the kth candidate solution in subpopulation i, $y_{ik}(s)$ is the sth solution variable of y_{ik}, μ_{ik}

and λ_{ik} are selection and replacing probabilities of solution y_{ik} respectively, σ_{ilmj} is the Euclidian distance between solution l in subpopulation i and solution j in subpopulation m, and N_i is the population size for subpopulation i.

Algorithm 2 – One generation of the proposed multi-population MOGA

```
Calculate the objective similarity between each pair of subpopulations using Algorithm 1
Calculate the rank of each solution y_ik in subpopulation i using non-dominated sorting method
Calculate selection probability μ_ik and replacing probability λ_ik based on the rank of solution y_ik
z ← y
For each subpopulation i
    /* within-subpopulation genetic operator */
  For each solution z_ik (k = 1 to N_i)
    For each solution variable s
      Based on roulette-wheel selection, use {μ_i} to probabilistically select y_ij in the same subpopulation i
      Perform global uniform crossover: z_ik(s) ← y_ij(s)
    Next variable
  Next solution
  /* cross-subpopulation migration */
  Find a suitable subpopulation m to pair with subpopulation i based on objective similarity levels
  For each solution z_ik (k = 1 to N_i)
    Calculate Euclidian distances { σ_ikml } between z_ik and each solution y_ml in subpopulation m
    Use λ_ik to probabilistically decide whether to replace z_ik
    If replacing then
      For each solution variable s
        Based on roulette-wheel selection, use Euclidian distances { σ_ikml } to probabilistically select the emigrating solution y_mj
        Perform migration: z_ik(s) ← y_mj(s)
      Next variable
    End if
    /* mutation */
    Probabilistically decide whether to mutate z_ik
  Next solution
Next subpopulation
y ← z
```

Based on the structure of the proposed multi-population MOGA, we see that the main difference between the proposed algorithm and the traditional multi-population MOGAs is the introduction of cross-subpopulation migration. An important aspect of multi-population methods is the communication between multiple subpopulations, which is configured by various parameters, including communication connection topologies, communication interval and communication rate. In many previous studies, these parameters need to be adjusted based on a priori knowledge of the optimization problem. However, it is hard to discover the best communication strategy in most real-world problems.

In the proposed algorithm, we use the migration method inspired by the ideas from biology to implement the communication between multiple subpopulations. That is, we firstly use MOGA to generate offspring solutions in each subpopulation, and then employ the cross-subpopulation migration to exchange their information. Therefore, it will often perform better due to the adaptive exchange of information between subpopulations. In this approach, the important characteristics of the problems including solution fitness, objective similarity and Euclidian distance are used in the cross-subpopulation migration to determine the communication parameter settings between subpopulations. It provides an information sharing mechanism to adaptively improve the entire population. This multi-population MOGA, which combines MOGA with the cross-population migration, can be treated as a template for designing other multi-population MOEAs. It has the common features of MOEAs, but also has the distinctive migration characteristics.

III. A Markov Chain Model of the Multi-Population MOGA

In the previous section, we introduced a multi-population MOGA. In this section, we establish a Markov model of the proposed multi-population MOGA, which can be studied as the functions of the proposed multi-population MOGA's tuning parameters to predict their impact on algorithmic performance, and find optimal values of the tuning parameters to realize real-time adaptation. Markov models have become a useful mathematical tool for EAs, including regular GAs [28, 29], BBO [30], and others [31]. In this paper, we will use a Markov model to produce insights to account for how well the proposed algorithm behaves.

Each Markov state in the multi-population MOGA is a specific population distribution. Each generation of the multi-population MOGA updates its population with a within-subpopulation genetic operator, and cross-subpopulation migration described in the previous section. A transition between states corresponds to the evolution of the population in one generation of the multi-population MOGA. So, in order to build a transition matrix, we need to model these operators in the multi-population MOGA.

Before developing the Markov model of the multi-population MOGA, we make three assumptions. First, a solution will not be replaced until the end of a generation. That is, the multi-population MOGA is generational rather than steady-state. This assumption guarantees that the selection and replacing probabilities remain the same throughout a given generation.

Second, a solution can replace itself. In other words, a solution crosses over itself in the within-subpopulation genetic operator. For cross-subpopulation migration, there is a chance that the replaced and emigrating solutions are the same.

Third, we use the preset selection and replacing probabilities for each solution rank rather than calculating them in each generation. All the ranks are calculated based on the classical non-dominated sorting method.

Assume that the multi-population MOGA consists of M subpopulations. We have a multi-objective optimization problem whose solution variables are binary. The bit number of a candidate solution in each subpopulation is the same, which is denoted as q. Use n to denote the cardinality of the search, that is, the total number of possible solutions in each subpopulation. Use $\overline{n}$ to denote the total number of possible solutions for all subpopulations. n and $\overline{n}$ are calculated by:

$$\begin{cases} n = 2^q \\ \overline{n} = n^M \end{cases} \tag{4}$$

x_j denotes the jth possible candidate solution in the search space of each subpopulation, and v_{ij} denotes the total number of possible solutions x_j in subpopulation i. Note that

$$\sum_{j=1}^{n} v_{ij} = N_i \tag{5}$$

The entire population in a multi-population MOGA can be generally represented as follows:

$$\begin{aligned} \text{Population} &= \left\{ \left[y_{11}, \cdots, y_{1N_1} \right], \left[y_{21}, \cdots, y_{2N_2} \right], \cdots, \left[y_{M1}, \cdots, y_{MN_M} \right] \right\} \\ &= \left\{ \begin{bmatrix} \underbrace{x_1, \cdots, x_1}_{v_{11}\text{ copies}}, \underbrace{x_2, \cdots, x_2}_{v_{12}\text{ copies}}, \cdots, \underbrace{x_n, \cdots, x_n}_{v_{1n}\text{ copies}} \end{bmatrix}, \right. \\ & \quad \begin{bmatrix} \underbrace{x_1, \cdots, x_1}_{v_{21}\text{ copies}}, \underbrace{x_2, \cdots, x_2}_{v_{22}\text{ copies}}, \cdots, \underbrace{x_n, \cdots, x_n}_{v_{2n}\text{ copies}} \end{bmatrix}, \\ & \quad \vdots \\ & \quad \left. \begin{bmatrix} \underbrace{x_1, \cdots, x_1}_{v_{M1}\text{ copies}}, \underbrace{x_2, \cdots, x_2}_{v_{M2}\text{ copies}}, \cdots, \underbrace{x_n, \cdots, x_n}_{v_{Mn}\text{ copies}} \end{bmatrix} \right\} \end{aligned} \tag{6}$$

where the y_{ij} solutions are ordered to group identical solutions. For each element of Equation (6), it is denoted in a compact format as follows:

$$y_{ik} = \begin{cases} x_1, \text{ when } k = 1, \cdots, v_{i1} \\ x_2, \text{ when } k = v_{i1} + 1, \cdots, v_{i1} + v_{i2} \\ \quad \vdots \\ x_n, \text{ when } k = \sum_{l=1}^{n-1} v_{il} + 1, \cdots, \sum_{l=1}^{n} v_{il} \end{cases} \tag{7}$$

Equation (7) can be rewritten as follows:

$$\begin{cases} y_{ik} = x_{z(k)} \\ z(k) = \min r, \text{ such that } \sum_{l=1}^{r} v_{il} > k \end{cases} \tag{8}$$

A. Within-subpopulation genetic operator

In Algorithm 2, we use roulette-wheel selection and global uniform crossover as a specific within-subpopulation genetic operator to produce an offspring. In [32], we have obtained the Markov model of this operator for a single population. Here we directly extend it for multiple subpopulations, and obtain the probability $P_{ikl}^{(1)}(v)$ that the kth solution in the ith subpopulation is equal to a given solution x_l at generation t+1, which is calculated as follows:

$$P_{ikl}^{(1)}(v) = \Pr\left(y_{ik,t+1} = x_l \right) = \prod_{s=1}^{q} \left[\frac{\sum_{j \in J_{il}(s)} v_{ij} \mu_{ij}}{\sum_{j=1}^{n} v_{ij} \mu_{ij}} \right] \tag{9}$$

where $P_{ikl}^{(1)}(v)$ is a function of the current population vector v at the tth generation (we will define the population vector precisely later, but for now we simply need to know that it represents the current population of the multi-population MOGA). The notation $J_{il}(s)$ in Equation (9) denotes the set of solution indices in subpopulation i that contains the same bit in position s as solution x_l. That is,

$$J_{il}(s) = \left\{ j : x_j(s) = x_l(s) \text{ in subpopulation } i \right\} \tag{10}$$

Furthermore, the value in brackets on the right hand side of Equation (9) denotes the probability of obtaining a certain bit at a certain position in a given solution, which is proportional to two factors: the total number of occurrences of that bit in the entire subpopulation, and the selection probabilities of the solutions that contain this bit.

B. Cross-subpopulation migration

The second part of the multi-population MOGA is cross-subpopulation migration, which is the migration between subpopulations. Considering the possibility of replacing a given solution in the cross-subpopulation migration, we have two possible scenarios: the first scenario is that the solution variables in the replaced solution won't be changed from generation t to t+1. We use $y_{ik}(s)$ to represent the sth bit in the kth solution in the ith subpopulation, which is represented as follows:

$$y_{ik}(s)_{t+1} = y_{ik}(s)_t = x_{k(z)}(s) \tag{11}$$

The second scenario is that migration is applied to the replaced solution. An important aspect that we need to consider here is how to compute the probability of the occurrence of each solution. For cross-subpopulation migration, we introduce Euclidian distances and use roulette-wheel selection based on Euclidian distance to select the emigrating solutions. The probability of obtaining a certain variable at a certain position in a given solution is proportional to the total number of the occurrences of that variable in the entire subpopulation and the replacing probabilities of the solutions that contain this variable. This probability is calculated for the sth component in the kth solution of the ith subpopulation as follows:

$$\Pr\left(y_{ik}(s)_{t+1} = x_l(s) \middle| \text{replacement} \right) = \frac{\sum_{j \in J_{il}(s)} v_{mj} \sigma_{ilmj}}{\sum_{j=1}^{n} v_{mj} \sigma_{ilmj}} \tag{12}$$

Considering both possibilities described above, given that the population vector at generation t is equal to v, the probability $P_{ikl}^{(2)}(v)$ that $y_{ik,t+1} = x_l$ at generation t+1 after cross-subpopulation migration can be calculated as

$$\begin{aligned} P_{ikl}^{(2)}(v) &= \Pr\left(y_{ik,t+1} = x_l \right) \\ &= \Pr(\text{no replacement}) \Pr\left(y_{ik,t+1} = x_l \middle| \text{no replacement} \right) + \\ &\quad \Pr(\text{replacement}) \Pr\left(y_{ik,t+1} = x_l \middle| \text{replacement from subpopulation } m \right) \\ &= \left(1 - \lambda_{iz(k)} \right) 1_{i0}\left(x_{z(k)} - x_l \right) + \lambda_{iz(k)} \prod_{s=1}^{q} \left[\frac{\sum_{j \in J_{il}(s)} v_{mj} \sigma_{ilmj}}{\sum_{j=1}^{n} v_{mj} \sigma_{ilmj}} \right] \end{aligned} \tag{13}$$

where 1_{i0} denotes the indicator function on $\{0\}$ in the subpopulation i. The first term of the right side of Equation (13) denotes the probability when replacement does not occur, and the second term on the right side of Equation (13) denotes the probability when replacement occurs by migration.

C. Combined within-subpopulation genetic operator and cross-subpopulation migration

Recall that Section II provides details of the multi-population MOGA. There are three steps in revising the population: within-subpopulation genetic operator, cross-subpopulation migration, and mutation. To find the total probability of obtaining a given solution, we combine the probabilities of these three operators. We combine the probabilities of the

within-subpopulation genetic operator and the cross-subpopulation migration to obtain $P_{ikl}^{(3)}(v)$:

$$P_{ikl}^{(3)}(v) = \sum_{j=1}^{n} P_{ikj}^{(1)}(v) P_{ijl}^{(2)}(v) \tag{14}$$

This is the probability that y_{ik} is equal to x_l in subpopulation i after both the within-subpopulation genetic operator and the cross-subpopulation migration have been considered.

D. Mutation

Mutation is another way to improve solutions in the multi-population MOGA. If we can obtain the probability of transforming a given solution to another given solution due to mutation, we combine this probability with Equation (14) to obtain the transition matrix for the Markov model of the multi-population MOGA.

Assuming that the mutation probability is predefined and constant, we can create a mutation matrix for each subpopulation. We use U_i to denote the mutation matrix for subpopulation i. We use U_{irl}, which is the lth element in the rth row in mutation matrix U_i, to denote the probability that solution x_r mutates to solution x_l in subpopulation i. Next, we combine U_i with Equation (14) to obtain the probability that $y_{ik,t+1} = x_l$ in subpopulation i after the within-subpopulation genetic operator, cross-subpopulation migration, and mutation, and it is calculated by

$$P_{ikl}^{(4)}(v) = \sum_{r=1}^{n} \sum_{j=1}^{n} P_{ikj}^{(1)}(v) P_{ijr}^{(2)}(v) U_{irl} \tag{15}$$

Now we extend the probability from the solution level to the population level. We introduce the term population vector, and use an example to illustrate it.

Example: Assume we have two subpopulations with four possible candidate solutions in subpopulation 1, and four possible candidate solutions in subpopulation 2. Then the population vector contains eight elements, which is shown in Figure 3. For example, a population vector [1 0 0 3 2 0 2 0] indicates that subpopulation 1 contains one S-11 and three S-14; and subpopulation 2 has two S-21 and two S-23.

Population vector							
S-11	S-12	S-13	S-14	S-21	S-22	S-23	S-24

Fig.3. Population vector in multi-population MOGA consisting of two subpopulations, where the number of possible candidate solutions in each subpopulation is four. The population vector has eight elements. S-*ik* represents the number of x_k solutions in subpopulation i.

Next the generalized multinomial theorem [33-34] is used to find the probability that population vector v transits to population vector u in subpopulation i after one generation. We use $\Pr_i(u|v)$ to denote this probability in subpopulation i:

$$\Pr_i(u|v) = \sum_{J \in Y_i} \prod_{k=1}^{N_i} \prod_{l=1}^{n} \left[P_{ikl}^{(4)}(v) \right]^{J_{ikl}},$$

$$\text{where } Y_i = \left\{ \begin{array}{l} J_i \in R^{N_i \times n} : J_{ikl} \in \{0,1\}, \\ \sum_{l=1}^{n} J_{ikl} = 1 \text{ for all } k, \sum_{k=1}^{N_i} J_{ikl} = u_{il} \text{ for all } l \end{array} \right\} \tag{16}$$

In Equation (16), $\Pr_i(u|v)$ is the element of the transition matrix P_i for subpopulation i, which represents the probability of transiting from one possible population vector to another. P_i is a $T_i \times T_i$ matrix, where T_i is the total number of possible population vectors in subpopulation i, which is calculated as follows [33]:

$$T_i = \left(n + N_i - 1,\ N_i \right) \tag{17}$$

That is, there are $T_i \times T_i$ combinations for u and v vectors in Equation (16). These $T_i \times T_i$ probabilities consist of the entries of the transition matrix P_i. Once having obtained the transition matrices P_i for each subpopulation, we can combine these matrices to form the transition matrix P for the multi-population MOGA. The matrix P can be calculated using the pseudo-code version of Algorithm 3. Note that the size of the matrix P is $T \times T$, where T is the total number of possible population vectors for the multi-population MOGA:

$$T = \prod_{i=1}^{M} T_i \tag{18}$$

Algorithm 3 – pseudo-code for the calculation of matrix *P*

```
For (t = 1; t ≤ T; t++)
    Set Count = 1;
    For (k1 = 1; k1 ≤ T1; k1++)
        For (k2 = 1; k2 ≤ T2; k2++)
                ⋮
            For (kM = 1; kM ≤ TM; kM++)
                P(Count, t) = P1(k1, t) P2(k2, t)…PM(kM, t);
                Count++;
            End for
        End for
    End for
End for
```

IV. Simulation Results

In this section, section (*A*) verifies the Markov model theory of the proposed multi-population MOGA with simulation results, and investigates the effects of cross-subpopulation migration on the proposed method. Sections (*B*) and (*C*) explore the effect of the subpopulation number and size on multi-population performance respectively, Section (*D*) compares the proposed multi-population MOEAs with the corresponding standard MOEAs on a set of multi-objective benchmark functions, and Section (*E*) discusses the benefits of the proposed multi-population method and the Markov model.

A. Theoretical Verification

In this section, we use two simple multi-objective problems to verify the Markov model derived in the previous section, which can exactly predict the steady state probability of each possible population vector in the multi-population MOGA. Meanwhile, we also use the Markov model to seek proper multi-population parameters to improve the optimization performance.

Suppose that the first multi-objective problem includes two cost functions. Each cost function contains two bits; that is, the possible solutions are {00, 01, 10, 11}. The cost functions are given as follows:

$$\left\{ \begin{array}{l} f_1 = x_1 + 2x_2 + 1 \\ f_2 = f_1 / (x_1 + x_2 + 1) + 1 \end{array} \right. \tag{19}$$

where f_1 is the first cost function, f_2 is the second cost function, x_1 is the first bit of a solution, and x_2 is the second bit of a solution. For this multi-objective problem, a smaller cost means better performance.

To justify the Markov model, we set two subpopulations for the multi-population MOGA, and perform Monte Carlo simulations to obtain the average performance. The simulation parameters include: 100 Monte Carlo simulations, 5000 generations for each Monte Carlo simulation, 4 individuals in each subpopulation, and the mutation probability of 0.01.

Furthermore, to explore the effect of multi-population on the MOGA performance, we investigate the cross-subpopulation migration. Based on Algorithm 2, there are two key tuning parameters for the cross-subpopulation migration: replacing probability λ and Euclidian distance σ respectively. In this experiment, the replacing probability is set to be linear and quadratic with respect to the solution rank respectively. Euclidian distance is set to be maximum, moderate, and minimum values respectively. Table I shows the Markov model and the simulated probabilities.

The cost functions shown in Equation (19) show that the population vector composed of Pareto solutions, called Pareto population vector, is [4 0 0 0 4 0 0 0], based on the population vector description presented in Figure 3. According to the results shown in Table I, first, we find that the theoretical results calculated by the Markov model match the simulation results for all the cases, which justifies the model derived in the previous section. For example, in the case of linear ranks and the maximum Euclidian distance, the probability of obtaining the Pareto population vector calculated by the Markov model is 0.6876, and the probability calculated by simulation is 0.6785.

Second, we witness that in the case of the same replacing probability, when the value of the Euclidian distance is the maximum, the probability of obtaining the Pareto population vector is the largest. When the value of the Euclidian distance is the minimum, the probability of obtaining the Pareto population vector is the smallest. For example, for the linear rank, the probability of obtaining the Pareto population vector calculated by the Markov model is 0.6876, 0.5785, 0.4868 for maximum, moderate, and minimum Euclidian distances respectively, which confirms that greater population diversity provides more opportunities to obtain a Pareto solution.

Third, we also find that in the case of the same Euclidian distance, the probability of obtaining the Pareto population vector for quadratic ranking is better than that for linear ranking. For example, for the maximum Euclidian distance, the probability of obtaining the Pareto population vector for quadratic ranks is 0.7806, and that for linear ranks is 0.6876, which confirms that replacing probability can affect the performance of the multi-population MOGA.

Finally, note that the CPU time for the calculation of the Markov model is 52 seconds, but the average CPU time for each simulation is 126 seconds. The proposed multi-population MOGA runs in MATLAB® on a 2.40 GHz Intel Pentium® 4 CPU with 4 GB of memory. The Markov model obtains more accurate probabilities than the simulation, and also does so with less CPU time. This is because Markov models are limited to problems with small population sizes and binary solution structures, which do not capture the characteristics of real-world problems.

Figure 4 shows typical simulation results for 5000 generations of the multi-population MOGA for the multi-objective problem in various combination cases. It is seen that all the Pareto population vectors agree with the results shown in Table I.

TABLE I THE FOUR MOST LIKELY POPULATIONS FOR DIFFERENT TUNING PARAMETERS FOR THE FIRST MULTI-OBJECTIVE PROBLEM. CPU TIMES FOR THE MARKOV MODEL AND THE SIMULATIONS ARE SHOWN IN THE LAST ROW OF THE TABLE.

Tuning parameters		Population Vector	Probability	
			Markov	Simulation
Replacing probability: Linear rank	Maximum Euclidian distance	4 0 0 0 4 0 0 0	0.6876	0.6785
		3 1 0 0 4 0 0 0	0.0969	0.0982
		4 0 0 0 3 1 0 0	0.0969	0.0981
		3 0 1 0 4 0 0 0	0.0321	0.0322
	Moderate Euclidian distance	4 0 0 0 4 0 0 0	0.5785	0.5714
		3 1 0 0 4 0 0 0	0.1617	0.1537
		4 0 0 0 3 1 0 0	0.0818	0.0913
		3 0 1 0 4 0 0 0	0.0393	0.0325
	Minimum Euclidian distance	4 0 0 0 4 0 0 0	0.4868	0.4783
		3 1 0 0 4 0 0 0	0.1370	0.1314
		4 0 0 0 3 1 0 0	0.1370	0.1365
		3 0 1 0 4 0 0 0	0.0385	0.0411
Replacing probability: Quadratic rank	Maximum Euclidian distance	4 0 0 0 4 0 0 0	0.7806	0.7764
		3 1 0 0 4 0 0 0	0.0649	0.0628
		4 0 0 0 3 1 0 0	0.0649	0.0628
		3 0 1 0 4 0 0 0	0.0321	0.0316
	Moderate Euclidian distance	4 0 0 0 4 0 0 0	0.7195	0.7074
		3 1 0 0 4 0 0 0	0.1118	0.1103
		4 0 0 0 3 1 0 0	0.0598	0.0657
		3 0 1 0 4 0 0 0	0.0339	0.0326
	Minimum Euclidian distance	4 0 0 0 4 0 0 0	0.6631	0.6617
		3 1 0 0 4 0 0 0	0.1033	0.1012
		4 0 0 0 3 1 0 0	0.1033	0.1016
		3 0 1 0 4 0 0 0	0.0313	0.0214
CPU time (s)			52	126

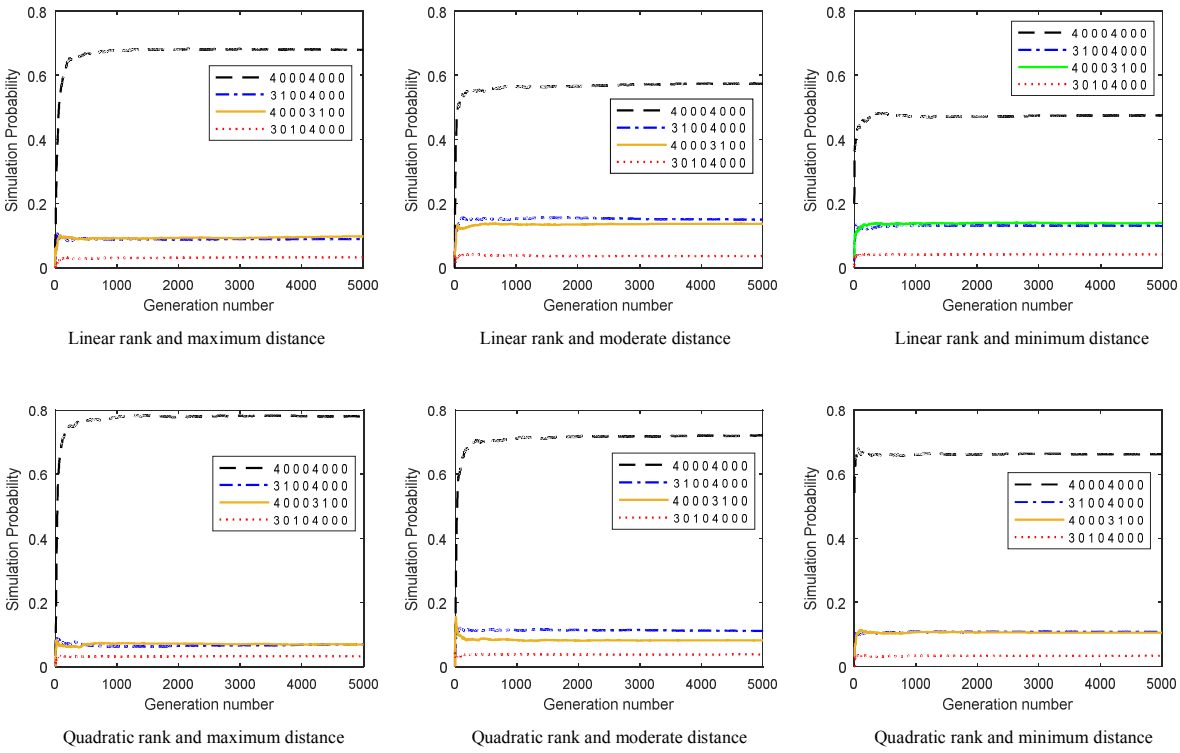

Fig.4. Typical multi-population MOGA simulation results for the first multi-objective problem in various combination cases, where the simulation probabilities of the four most probable population vectors are shown.

The second multi-objective problem includes three cost functions. Each cost function includes two bits, and the possible solutions are {00, 01, 10, 11}. The cost functions are:

$$\begin{cases} f_1 = 2x_1 + x_2 + 1 \\ f_2 = (f_1 + 1)/(x_1 + x_2 + 1) + 1 \\ f_3 = (x_1 + x_2)/(f_1 + 1) + 1 \end{cases} \tag{20}$$

where f_1 is the first cost function, f_2 is the second cost function, f_3 is the third cost function, x_1 is the first bit of a solution, and x_2 is the second bit of a solution.

For the second multi-objective problem, we set three subpopulations; the other parameter settings are the same as for the first multi-objective problem. As in the first problem, a smaller cost means better performance, but it is more complex due to the exponential increase of matrix sizes with the problem size and subpopulation number. Table II shows comparisons between theoretical (Markov) and simulated results.

The cost functions in Equation (20) show that the population vector composed of Pareto solutions is [4 0 0 0 4 0 0 0 4 0 0 0] based on the description of the population vector in Figure 3. According to the results in Table II, we observe that the theoretical results calculated by the Markov model match well the simulation results for all the cases, which again confirms validity of the derived Markov model. Meanwhile, we also find that the conclusions about replacing probability and Euclidian distance are in accord with those obtained from the first multi-objective problem, which again confirms the important effect of cross-subpopulation migration on the optimization performance of the multi-population MOGA.

In addition, for this multi-objective problem, the CPU time for the Markov model is 947 seconds, but the average CPU time of each simulation is 224 seconds. In this case, the computational effort of the Markov model is higher than that of the simulation. This is because the total number of possible population vectors increases exponentially with the number of subpopulations as seen in Equations (17) and (18), leading to a large transition matrix that is difficult to handle.

B. Effect of subpopulation number

Subpopulation number is one of the important tuning parameters in the multi-population method. To explore its effect on the proposed algorithm, we consider four cases with one, two, three and four subpopulations. For migration operators, we set the replacing probability to be linear with respect to the solution rank, and Euclidian distance to be maximum value. Other parameters are the same as those used in the previous experiment. Table III shows the Markov model and simulated probabilities for the different subpopulation numbers for two multi-objective problems.

According to Table III, on the one hand, we find that the probability of obtaining the Pareto population vector for many subpopulation numbers is better than that for a few subpopulation numbers. For example, for one subpopulation, the probability is 0.6307 for the first problem, and is 0.7291 for four subpopulations, which confirms that the subpopulation number can affect the performance of the multi-population MOGA. On the other hand, for the problems we study, when the subpopulation number increases, the effect becomes less clear. For example, for the second problem, the probability is 0.6342 for three subpopulations, and 0.6419 for four subpopulations. This shows that for the multi-population

method, the appropriate subpopulation number is important, which depends on the problems we study.

TABLE II THE FOUR MOST LIKELY POPULATIONS FOR DIFFERENT TUNING PARAMETERS FOR THE SECOND MULTI-OBJECTIVE PROBLEM. CPU TIMES FOR THE MARKOV MODEL AND THE SIMULATIONS ARE SHOWN IN THE LAST ROW OF THE TABLE.

Tuning parameters		Population Vector	Probability	
			Markov	Simulation
Replacing probability: Linear rank	Maximum Euclidian distance	4 0 0 0 4 0 0 0 4 0 0 0	0.6342	0.6311
		3 1 0 0 4 0 0 0 4 0 0 0	0.0813	0.0809
		4 0 0 0 4 0 0 0 3 1 0 0	0.0813	0.0810
		4 0 0 0 3 0 1 0 4 0 0 0	0.0395	0.0475
	Moderate Euclidian distance	4 0 0 0 4 0 0 0 4 0 0 0	0.5315	0.5284
		3 1 0 0 4 0 0 0 4 0 0 0	0.1562	0.1497
		4 0 0 0 4 0 0 0 3 1 0 0	0.1562	0.1497
		4 0 0 0 3 0 1 0 4 0 0 0	0.0321	0.0419
	Minimum Euclidian distance	4 0 0 0 4 0 0 0 4 0 0 0	0.4452	0.4387
		3 1 0 0 4 0 0 0 4 0 0 0	0.1341	0.1315
		4 0 0 0 4 0 0 0 3 1 0 0	0.1341	0.1285
		4 0 0 0 3 0 1 0 4 0 0 0	0.0396	0.0412
Replacing probability: Quadratic rank	Maximum Euclidian distance	4 0 0 0 4 0 0 0 4 0 0 0	0.7245	0.7213
		3 1 0 0 4 0 0 0 4 0 0 0	0.0421	0.0413
		4 0 0 0 4 0 0 0 3 1 0 0	0.0421	0.0501
		4 0 0 0 3 0 1 0 4 0 0 0	0.0295	0.0310
	Moderate Euclidian distance	4 0 0 0 4 0 0 0 4 0 0 0	0.6684	0.6605
		3 1 0 0 4 0 0 0 4 0 0 0	0.1013	0.0944
		4 0 0 0 4 0 0 0 3 1 0 0	0.0496	0.0516
		4 0 0 0 3 0 1 0 4 0 0 0	0.0301	0.0322
	Minimum Euclidian distance	4 0 0 0 4 0 0 0 4 0 0 0	0.6035	0.6003
		3 1 0 0 4 0 0 0 4 0 0 0	0.0923	0.0815
		4 0 0 0 4 0 0 0 3 1 0 0	0.0923	0.0876
		4 0 0 0 3 0 1 0 4 0 0 0	0.0287	0.0312
CPU time (s)			947	224

C. *Effect of subpopulation size*

Subpopulation size is another tuning parameter in the multi-population method. To explore the effect of the subpopulation size, we consider three cases of two, four and eight individuals in each subpopulation. In this experiment, we set two subpopulations for the first multi-objective problem, and three subpopulations for the second problem. Other parameters are the same as those used in the previous experiments. Table IV shows the Markov model and simulated probabilities for the different subpopulation sizes for two multi-objective problems.

According to Table IV, we observe that the probability of obtaining the Pareto population vector is the largest for the four individuals. For example, for the first problem, the probability is 0.5724, 0.6876 and 0.6037 for two, four and eight individuals respectively. This experiment shows that for the multi-population method, the appropriate subpopulation size is important, which depends on the problems we study.

D. *Optimization for Benchmarks*

This subsection presents multi-objective unconstrained functions from the 2009 IEEE Congress on Evolutionary Computation (CEC) to show the results of the proposed multi-population method on MOEAs. These functions are briefly summarized in Table V, and the complete definition of each function is available in the literature [35].

In this experiment, we integrate the cross-subpopulation migration into the established multi-objective biogeography-based optimization (MBBO) [36], multi-objective evolutionary algorithm based on decomposition (MOEA/D) [37], and multiple trajectory search (MTS) algorithm [38] under the condition of the original algorithm. Here, we choose MBBO because it is one of the most recent MOEAs. We choose MOEA/D because it is one of the most popular and outstanding MOEAs, and we choose MTS because it is one of the most successful MOEAs reported in the CEC competition. For the multi-population versions of these algorithms, we use the quadratic rank as the replacing probability. Meanwhile, for solving real-world problems, to improve Euclidian distance based schemes, we also employ a simple multi-population strategy: For similar subpopulations during the optimization, we firstly preserve one subpopulation and delete the other similar subpopulations. We then create the same number of new subpopulations, which consist of the following candidate solutions: 1/3 of them are random copies from the preserved subpopulation, 1/3 are from the best solutions in the entire population, and 1/3 are randomly selected from the other subpopulations.

In this experiment, for the four presented algorithms, subpopulation number is set to 4 based on the results in the previous experiments. The population size of each subpopulation is 50, so the entire population size is 200. To obtain fair comparisons, a population size for the standard versions of these algorithms is set to 200. For the multi-population MOEA/D, we use the same weights for these subpopulations, and other parameter settings of these algorithms are referred to the original references [36-38]. The benchmark functions are compared by implementing discretized coding of all the standard and multi-population algorithms. The granularity or precision of each solution variable is set to 0.01 to provide for reasonable computational effort. Note that benchmark tests might result in a different conclusion if we use a different setup or if we change

precision. Since the precision of each solution variable is set to a small value, we expect real coding to improve performance by 2%-3% in theory. We evaluate each algorithm with 10 Monte Carlo simulations and 10,000 generations for each Monte Carlo simulation. Figures S1, S2 and S3 in this paper's supplemental material show the approximate Pareto front graphs obtained by the different algorithms. Note that due to space limitations, we show only the representative two-objective benchmark functions U02, U04 and three-objective benchmark function U08. It is observed that Pareto fronts of the multi-population MOGA, MBBO, MOEA/D, and MTS are closer to ideal Pareto fronts than those of the corresponding standard algorithms.

Furthermore, we use hypervolume as the performance metric, which denotes the volume of the objective space between the obtained solution set and the reference point. It also gives the solution set a comprehensive assessment with respect to convergence and diversity. Similar to [39], the reference point is set to 1.1 times the upper bound of the Pareto front. For a minimization problem, a smaller hypervolume indicates a better Pareto front approximation. The computational details of hypervolume can be found in the literature [13]. Table VI shows the average performance of multi-population MOGA, MBBO, MOEA/D, and MTS, and their standard versions with respect to the hypervolume metric.

The results shown in Table VI are divided into standard MOGA versus multi-population MOGA, standard MBBO versus multi-population MBBO, standard MOEA/D versus multi-population MOEA/D, and standard MTS versus multi-population MTS. From Table VI, we observe that multi-population MOGA performs better than the standard MOGA, multi-population MBBO better than standard MBBO, multi-population MOEA/D better than standard MOEA/D, and multi-population MTS better than standard MTS, on all the benchmark functions. That is, the multi-population versions of these algorithms are significantly better than their standard versions. It also shows that the proposed multi-population method is an effective method to improve the optimization performance of the MOEAs. The reason for its improved performance is that multi-population MOEAs effectively use cross-subpopulation migration that can significantly increase the communication between the subpopulations to improve the population diversity, compared to traditional MOEAs.

The average running time of these algorithms is shown in the last row of Table VI. From the table, we see that the average running time of the multi-population versions of MOGA, MBBO, MOEA/D, and MTS is much less than their standard versions. The reason is that the multi-population algorithms use the multiple parallel subpopulations of a relatively small size, while the standard algorithms use a single population that is four times as large as their multi-populations. Based on this justification, we believe that multiple subpopulations can be seen as parallel processing, which can reduce the computational effort.

TABLE III THE FOUR MOST LIKELY POPULATIONS FOR THE CASES OF ONE, TWO, THREE AND FOUR SUBPOPULATIONS.

Tuning parameters		Population Vector	Probability	
			Markov	Simulation
First problem	One subpopulation	4 0 0 0	0.6307	0.6259
		3 1 0 0	0.1235	0.1211
		3 0 1 0	0.1235	0.1206
		3 0 0 1	0.0393	0.0378
	Two subpopulations	4 0 0 0 4 0 0 0	0.6876	0.6785
		3 1 0 0 4 0 0 0	0.0969	0.0982
		4 0 0 0 3 1 0 0	0.0969	0.0981
		3 0 1 0 4 0 0 0	0.0321	0.0322
	Three subpopulations	4 0 0 0 4 0 0 0 4 0 0 0	0.7239	0.7168
		3 1 0 0 4 0 0 0 4 0 0 0	0.0861	0.0822
		4 0 0 0 4 0 0 0 3 1 0 0	0.0861	0.0813
		4 0 0 0 3 0 1 0 4 0 0 0	0.0245	0.0242
	Four subpopulations	4 0 0 0 4 0 0 0 4 0 0 0 4 0 0 0	0.7291	0.7236
		3 1 0 0 4 0 0 0 4 0 0 0 4 0 0 0	0.0823	0.0815
		4 0 0 0 4 0 0 0 3 1 0 0 4 0 0 0	0.0823	0.0803
		4 0 0 0 4 0 0 0 4 0 0 0 3 0 1 0	0.0271	0.0232
Second problem	One subpopulation	4 0 0 0	0.5216	0.5101
		3 1 0 0	0.1517	0.1512
		3 0 1 0	0.1517	0.1520
		3 0 0 1	0.0395	0.0347
	Two subpopulations	4 0 0 0 4 0 0 0	0.5842	0.5792
		3 1 0 0 4 0 0 0	0.1225	0.1149
		4 0 0 0 3 1 0 0	0.1225	0.1187
		3 0 1 0 4 0 0 0	0.0332	0.0322
	Three subpopulations	4 0 0 0 4 0 0 0 4 0 0 0	0.6342	0.6311
		3 1 0 0 4 0 0 0 4 0 0 0	0.0813	0.0809
		4 0 0 0 4 0 0 0 3 1 0 0	0.0813	0.0810
		4 0 0 0 3 0 1 0 4 0 0 0	0.0395	0.0475
	Four subpopulations	4 0 0 0 4 0 0 0 4 0 0 0 4 0 0 0	0.6419	0.6447
		3 1 0 0 4 0 0 0 4 0 0 0 4 0 0 0	0.0784	0.0723
		4 0 0 0 4 0 0 0 3 1 0 0 4 0 0 0	0.0784	0.0708
		4 0 0 0 4 0 0 0 4 0 0 0 3 0 1 0	0.0371	0.0324

TABLE IV THE FOUR MOST LIKELY POPULATIONS FOR THE CASES OF TWO, FOUR AND EIGHT INDIVIDUALS IN EACH SUBPOPULATION

Tuning parameters		Population Vector	Probability	
			Markov	Simulation
First problem	Two individuals	4 0 0 0 4 0 0 0	0.5724	0.5672
		3 1 0 0 4 0 0 0	0.1423	0.1394
		4 0 0 0 3 1 0 0	0.1423	0.1394
		3 0 1 0 4 0 0 0	0.0396	0.0371
	Four individuals	4 0 0 0 4 0 0 0	0.6876	0.6785
		3 1 0 0 4 0 0 0	0.0969	0.0982
		4 0 0 0 3 1 0 0	0.0969	0.0981
		3 0 1 0 4 0 0 0	0.0321	0.0322
	Eight individuals	4 0 0 0 4 0 0 0	0.6037	0.5934
		3 1 0 0 4 0 0 0	0.1250	0.1174
		4 0 0 0 3 1 0 0	0.1250	0.1203
		3 0 1 0 4 0 0 0	0.0411	0.0402
Second problem	Two individuals	4 0 0 0 4 0 0 0	0.4934	0.4816
		3 1 0 0 4 0 0 0	0.1672	0.1605
		4 0 0 0 3 1 0 0	0.1672	0.1589
		3 0 1 0 4 0 0 0	0.0510	0.0486
	Four individuals	4 0 0 0 4 0 0 0	0.5842	0.5792
		3 1 0 0 4 0 0 0	0.1225	0.1149
		4 0 0 0 3 1 0 0	0.1225	0.1187
		3 0 1 0 4 0 0 0	0.0332	0.0322
	Eight individuals	4 0 0 0 4 0 0 0	0.5136	0.5046
		3 1 0 0 4 0 0 0	0.1527	0.1473
		4 0 0 0 3 1 0 0	0.1527	0.1405
		3 0 1 0 4 0 0 0	0.0401	0.0355

E. Discussion

Multi-population methods have been shown to be a simple and effective means for enhancing optimization performance. In this work, we embed multi-population methods into evolutionary algorithms to solve various MOPs. From the experimental results, we can summarize that

(1) The proposed multi-population approach is effective and efficient. It takes full advantage of cross-subpopulation as the communication mechanism between subpopulations to increase population diversity.

(2) The overall performance of the proposed multi-population MOEAs on a set of multi-objective benchmarks is superior to or comparable with the corresponding single-population versions.

(3) Simulation studies show that the Markov model is still an excellent theoretical tool to reveal the algorithmic characteristics and justify the algorithm performance.

(4) The Markov model of the proposed multi-population MOGA confirms the important effect of cross-subpopulation migration on the optimization performance. When the replacing probability is set to be quadratic, and the Euclidian distance is the maximum, the probability of obtaining the Pareto population vector is the largest.

(5) The Markov model of the proposed multi-population MOGA also confirms that the appropriate subpopulation number and subpopulation size are important, and these depend on the problems we study.

TABLE V CEC 2009 UNCONSTRAINED MULTI-OBJECTIVE BENCHMARK FUNCTIONS, WHERE n_0 DENOTES THE NUMBER OF DIMENSIONS IN EACH OBJECTIVE.

Function	Num. of objectives	Search space
U01: unconstrained problem 1	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U02: unconstrained problem 2	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U03: unconstrained problem 3	2	$[0,1]^{n_0}$, $n_0=30$
U04: unconstrained problem 4	2	$[0,1]\times[-2,2]^{n_0-1}$, $n_0=30$
U05: unconstrained problem 5	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U06: unconstrained problem 6	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U07: unconstrained problem 7	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U08: unconstrained problem 8	3	$[0,1]^2\times[-2,2]^{n_0-2}$, $n_0=30$
U09: unconstrained problem 9	3	$[0,1]^2\times[-2,2]^{n_0-2}$, $n_0=30$
U10: unconstrained problem 10	3	$[0,1]^2\times[-2,2]^{n_0-2}$, $n_0=30$

TABLE VI OPTIMIZATION RESULTS OF MULTI-POPULATION MOGA, MBBO, MOEA/D, AND MTS, AND THEIR STANDARD VERSIONS FOR 10 UNCONSTRAINED MULTI-OBJECTIVE BENCHMARK FUNCTIONS. AVERAGE CPU TIMES (MINUTE) ARE SHOWN IN THE LAST ROW OF THE TABLE.

Function	Algorithms							
	S-MOGA	M-MOGA	S-MBBO	M-MBBO	S-MOEA/D	M-MOEA/D	S-MTS	M-MTS
U01	62.32	41.36	70.19	44.57	57.21	40.55	53.73	39.15
U02	44.75	21.35	41.35	19.23	37.78	17.23	38.12	19.52
U03	315.4	213.7	328.4	226.5	284.5	207.9	256.3	201.2
U04	11.25	8.311	9.925	8.247	9.252	7.923	9.361	7.640
U05	293.6	127.5	274.3	115.2	157.3	118.4	192.5	108.3
U06	703.6	434.2	657.1	463.2	496.2	402.1	452.3	399.5
U07	67.42	42.63	68.71	42.39	45.66	39.87	50.12	41.24
U08	726.1	533.4	682.7	503.5	587.8	455.6	600.4	474.0
U09	2135.6	1525.4	2347.5	1432.0	1656.1	1386.2	1688.9	1389.6
U10	3133.2	2155.5	3242.1	1943.2	2301.2	1768.4	2621.5	1805.0
Time (Min.)	578.2	344.6	567.5	332.7	254.7	212.2	368.6	301.5

V. CONCLUSIONS

Multi-population is an effective optimization component often used in evolutionary algorithms in order to improve performance. In this paper, we first proposed a new multi-population method used in MOGA, which includes the within-subpopulation genetic operator, and cross-subpopulation migration. Then we derived a Markov model of the proposed multi-population MOGA. The model outlines the theoretical probability of each possible population occurring simultaneously with multiple objectives as the generation number goes to infinity. It has been confirmed through simulation that the proposed multi-population method can enhance the exploration ability of MOGA. Finally, the proposed multi-population method is applied to other popular MOEAs, and they are investigated on a set of the CEC multi-objective benchmarks. The empirical results show that multi-population MOEAs can obtain better optimization performance than their corresponding single-population version for the multi-objective benchmarks we study.

In future research, at least three directions may be considered. First, the proposed multi-population method is combined with MOEAs, and improves the multi-objective optimization performance. The multi-population framework presented here can be extended for other types of optimization algorithms, for example, dynamic optimization, large-scale optimization and complex system optimization. Second, we intend to design a method to further reduce computational effort during the Markov model development. As seen in Equations (17) and (18), the model is computationally expensive because of the size of the Markov transition matrix. This limitation could be partially addressed by grouping similar Markov model states together into a single state [27]. Finally, solving real-world multi-objective application problems is the ultimate goal of the population-based algorithms, and it would be useful to apply the proposed multi-population MOEAs to various real-world problems.

REFERENCES

[1] H. Li, Q. Zhang, J. Deng, "Biased multiobjective optimization and decomposition algorithm," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 52-66, Jan. 2017.

[2] S. Gee, K. C. Tan, H. Abbass, "A benchmark test suite for dynamic evolutionary multiobjective optimiztion," *IEEE Trans. Cybern.*, vol. 47, no. 2, pp. 461-472, Feb. 2017.

[3] D. Simon, *Evolutionary Optimization Algorithms*, John Wiley & Sons, Hoboken, NJ, 2013.

[4] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, Q. Zhang, "Multi-objective evolutionary algorithms: A survey of the state of the art," *Swarm. Evol. Comput*., vol. 1, no. 1, pp. 32-49, 2011.

[5] S. Jiang, S. Yang, "An improved multiobjective optimization evolutionary algorithm based on decomposition for complex Pareto fronts," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 421-437, Feb. 2016.

[6] C. A. C. Coello, G. T. Pulido, M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256-279, June. 2004.

[7] H. Wang, Q. Zhang, L. Jiao, X. Yao, "Regularity model for noisy multiobjective optimization," *IEEE Trans. Cybern.*, vol. 46, no. 9, pp. 1997-2009, Sep. 2016.

[8] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182-197, Dec. 2002.

[9] Q. Zhang, H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712-731, Dec. 2006.

[10] H. Kwasnicka, M. Przewozniczek, "Multi population pattern searching algorithm: a new evolutionary method based on the idea of messy genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 715-734, Oct. 2011.

[11] C. Li, T. T. Nguyen, M. Yang, S. Yang, S. Zeng, "Multi-population methods in unconstrained continuous dynamic environments: The challenges," *Inform. Sci*., vol. 296, pp. 95-118, 2015.

[12] L. Chen, F. J. Chang, "Applying a real-coded multi-population genetic algorithm to multi-reservoir operation," *Hydrol. Process*., pp. 688-698, 2007.

[13] H. Ma, D. Simon, *Evolutionary Computation with Biogeography-based Optimization*, John Wiley & Sons, Hoboken, NJ, 2017.

[14] J. K. Cochran, S. Horng, J. W. Fowler, "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines", *Comput. Opera. Res*., vol. 30, pp. 1087-1102, 2003.

[15] M. zandieh, N. Karimi, "An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times," *J. Intell, Manufact*., vol. 22, no. 6, pp. 979-989, 2011.

[16] P. Kersting, A. Zabel, "Optimizing NC-tool paths for simultaneous five-axis milling based on multi-population multi-objective evolutionary algorithms," *Adv. Eng. Softw*., vol. 40, no. 6, pp. 452-463, 2009.

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

[18] J. Zheng, C. Chien, M. Gen, "Multi-objective multi-population biased random-key genetic algorithm for the 3-D container loading problem," *Comput. Indust. Engin*., vol. 89, pp. 80-87, 2015.

[19] X. Wang, L. Tang, "An adaptive multi-population differential evolution algorithm for continuous multi-objective optimization," *Inform. Sci*., vol. 348, pp. 124-141, 2016.

[20] C. Li, T. Nguyen, M. Yang, M. Mavrovouniotis, S. Yang, "An adaptive multipopulation framework for locating and tracking multiple optima," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 590-605, Aug. 2016.

[21] D. Du, D. Simon, "Complex system optimization using biogeography-based optimization," *Math. Prob. Engin*., vol. 2013, pp. 1-19, 2013.

[22] W. Sheng, S. Chen, M. Sheng, G. Xiao, J. Mao, Y. Zheng, "Adaptive multisubpopulation competition and multiniche crowding-based memetic algorithm for automatic data clustering," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 838-858, Dec. 2016.

[23] H. Cheng, S. Yang, X. Wang, "Immigrants-enhanced multi-population genetic algorithms for dynamic shortest path routing problems in mobile ad hoc networks," *Appl. Artifi. Intell.*, vol. 26, pp. 673-695, 2012.

[24] C. Erick, "Topologies, migration rates, and multi-population parallel genetic algorithms," Proceeding of *1st Conf. Gen. Evol. Comput.*, Florida, USA, pp. 13-17, 1999.

[25] R. MacArthur, E. Wilson, *The Theory of Island Biogeography*, Princeton University Press, New Jersey, 1967.

[26] T. Lenton, "Gaia and natural selection," *Nature*, vol. 394, no. 6692, 439-447, 1998.

[27] C. Reeves, J. Rowe, *Genetic Algorithms: Principles and Perspectives*. Norwell, MA: Kluwer, 2003.

[28] J. Suzuki, "A Markov chain analysis on simple genetic algorithms," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 25, no. 4, pp. 655-659, Apr. 1995.

[29] A. Wright, Y. Zhao, "Markov chain models of genetic algorithms," *In Proc. Gen. Evolut. Comput. Conf.*, vol. 1, pp. 734-741, 1999.

[30] D. Simon, M. Ergezer, D. Du, R. Rarick, "Markov models for biogeography-based optimization," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 41, no. 1, pp. 299-306, Feb. 2011.

[31] J. Suzuki, "A further result on the Markov chain model of genetic algorithms and its application to a simulated annealing-like strategy," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 28, no. 1, pp. 95-102, Feb. 1998.

[32] D Simon, R Rarick, M Ergezer, D Du, "Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms," *Inform. Sci.*, vol. 181, no. 7, pp. 1224-1248, 2011.

[33] N. Beaulieu, "On the generalized multinomial distribution, optimal multinomial detectors, and generalized weighted partial decision detectors," *IEEE Trans. Commun.*, vol. 39, no. 2, pp. 193-194, Feb. 1991.

[34] C. Grinstead, J. Snell, *Introduction to Probability*. American Mathematical Society, 1998.

[35] Q. Zhang, A. Zhou, S. Z. Zhao, P. N. Suganthan, W. Liu, S. Tiwari, "Multi-objective optimization test instances for the CEC 2009 special session and competition," *Technical Report*, 2008.

[36] H. Ma, S. Su, D. Simon, M. Fei, "Ensemble multi-objective biogeography-based optimization with application to automated warehouse scheduling," *Engin. Appli. Artifi. Intell.*, vol. 44, pp. 79-90, 2015.

[37] C. Chen, Y. Chen, Q. Zhang, "Enhancing MOEA/D with guided mutation and priority update for multi-objective optimization," In: *Proc. IEEE Congress Evol. Comput.*, Trondheim, Norway, pp. 209-216, 2009.

[38] L. Tseng, C. Chen, "Multiple trajectory search for unconstrained/constrained multi-objective optimization," In: *Proc. IEEE Congress Evol. Comput.*, Trondheim, Norway, pp. 1951-1958, 2009.

[39] H. Ishibuchi, Y. Hitotsuyanagi, N. Tsukamoto, Y. Nojima, "Many-objective test problems to visually examine the behavior of multi-objective evolution in a decision space," In: *Proc. Intern. Conf. Paral. Prob. Solv. Nature*, Krakow, Poland, pp. 91-100, 2010.

Haiping Ma received the B. S. degree from Shaoxing University, Shaoxing, China, the M. S. degree from the Taiyuan University of Technology, Taiyuan, China, and the Ph.D. degree from Shanghai University, Shanghai, China, in 2004, 2007, and 2014, respectively, all in control theory and control engineering. He is currently a visiting scholar with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK, and an Associate Professor with the College of Mathematics, Physics and Information, Shaoxing University. In 2015, he received the Outstanding Ph.D. Dissertation Award, Chinese Association of System Simulation, China. He has published over 30 research papers on evolutionary algorithms and applications. His current research interests include evolutionary computation, information fusion, and intelligent control. He is the author of the textbook *Evolutionary Computation with Biogeography-based Optimization* (John Wiley & Sons, 2017).

Minrui Fei received his B. S. and M. S. from Shanghai University of Technology, and his Ph.D. from Shanghai University, all in control theory and control engineering. He is presently a professor and doctoral supervisor of Shanghai University since 1998. His teaching and research interests include intelligent control, complex system modeling, networked control systems and evolutionary computation. He has published over 190 research papers and co-edited seven conference proceedings in his field. He is a director at the Shanghai Key Laboratory of Power Station Automation Technology, Shanghai University, vice-chairman of Chinese Association for System Simulation, standing director of China Instrument and Control Society, and director of Chinese Artificial Intelligence Association. His research has been funded by the National Science Foundation and several industrial organizations.

Zheheng Jiang received the B. S. degree in Electrical Engineering and Automation (Grid Monitoring) from Nanjing Institute of Technology and the M. S. degree in Software Development from Queen's University Belfast, U.K. He is currently pursuing his Ph.D. degree with Department of Informatics, University of Leicester, U.K. His current research interests include machine learning for vision, object detection and recognition, video analysis and event recognition.

Ling Li received her Ph.D. degree from Imperial College London. She is currently the Director of Internationalization at the School of Computing, University of Kent. And she is also the founding coordinator of Laboratory of Brain Cognition Computing (BC2 Lab) of the school, responsible for coordinating multidisciplinary research between Computing, Sports and local NHS hospitals. Before she joined the University of Kent, she had six-year research experience at Imperial College London with a focus on understanding body sensor data (EEG, EMG, ECG, eAR-sensor, and etc.). Her current research interests include adaptive filtering, computational intelligence, and machine learning methods for pattern classification. She now serves at the editorial board of Brain Informatics and the secretary of IEEE Computing Society in UK and Ireland.

Huiyu Zhou received a Bachelor of Engineering degree in Radio Technology from the Huazhong University of Science and Technology of China, and a Master of Science degree in Biomedical Engineering from the University of Dundee of United Kingdom, respectively. He was then awarded a Doctor of Philosophy degree in Computer Vision from the Heriot-Watt University, Edinburgh, United Kingdom. Dr. Zhou is Reader at Department of Informatics, University of Leicester, United Kingdom. He has published over 150 peer-reviewed papers in the field. His research work has been or is being supported by UK EPSRC, EU, Royal Society, Leverhulme Trust, Puffin Trust, Invest NI and industry.

Danny Crookes received a BSc and PhD in 1977 and 1980 respectively. He was appointed to the Chair of Computer Engineering in 1993 at Queen's University Belfast, and was Head of Computer Science from 1993-2002. He was Director of Research for Speech, Image and Vision Systems at the Institute for Electronics, Communications and Information Technology (ECIT) at Queen's University Belfast. His research interests include medical image processing, and speech enhancement and separation. He has published over 230 scientific papers in journals and international conferences.