Natural Computing (2018) 17:611–639
https://doi.org/10.1007/s11047-018-9670-5

Balanced-evolution genetic algorithm for combinatorial optimization problems: the general outline and implementation of balanced-evolution strategy based on linear diversity index

HongGuang Zhang[1] · YuanAn Liu[1] · Jie Zhou[1]

Published online: 30 January 2018

Abstract

How to rationally inject randomness to control population diversity is still a difficult problem in evolutionary algorithms. We propose balanced-evolution genetic algorithm (BEGA) as a case study of this problem. Similarity guide matrix (SGM) is a two-dimensional matrix to express the population (or subpopulation) distribution in coding space. Different from binary-coding similarity indexes, SGM is able to be suitable for binary-coding and symbol-coding problems, simultaneously. In BEGA, opposite-direction and forward-direction regions are defined by using two SGMs as reference points, respectively. In opposite-direction region, diversity subpopulation always tries to increase Hamming distances between themselves and the current population. In forward-direction region, intensification subpopulation always tries to decrease Hamming distances between themselves and the current elitism population. Thus, diversity subpopulation is more suitable for injecting randomness. Linear diversity index (LDI) measures the individual density around the center-point individual in coding space, which is characterized by itself linearity. According to LDI, we control the search-region ranges of diversity and intensification subpopulations by using negative and positive perturbations, respectively. Thus, the search efforts between exploration and exploitation are balanced. We compared BEGA with CHC, dual-population genetic algorithm, variable dissortative mating genetic algorithm, quantum-inspired evolutionary algorithm, and greedy genetic algorithm for 12 benchmarks. Experimental results were acceptable. In addition, it is worth noting that BEGA is able to directly solve bounded knapsack problem (i.e. symbol-coding problem) as one EA-based solver, and does not transform bounded knapsack problem into an equivalent binary knapsack problem.

Keywords Population diversity control · Feedback control scheme · Similarity guide matrix · Linear diversity index · Symbol-coding problem · Bounded knapsack problem

1 Introduction

Evolutionary algorithms (EAs) are widely used as global optimizers, such as genetic algorithms (GAs) (De Jong 1975; Holland 1975) and simulated annealing (Dekkers and Aarts 1991). Population diversity control is important to obtain desired solutions in each EA. However, how to rationally inject randomness to control population diversity is still a difficult problem. In the above problem, there are two sub-problems: (1) How to determine which subpopulations are more suitable for injecting randomness (i.e. first sub-problem)? (2) How much randomness is injected is suitable for population diversity control (i.e. second sub-problem)?

✉ HongGuang Zhang
hongguang-zhang@bupt.edu.cn; hongguang-zhang@163.com

YuanAn Liu
yuliu@bupt.edu.cn

Jie Zhou
zhoujie-1115@163.com

[1] Beijing Key Laboratory of Work Safety Intelligent Monitoring, School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

We propose balanced-evolution genetic algorithm (BEGA). BEGA realizes the balanced evolution of the population, which is controlled by population diversity measures. BEGA is characterized by this balanced-evolution strategy. From a coding-space perspective, this is a case study for solving first and second sub-problems. Note that, coding space is also called genotypic space that is related to coding type and coding position, and solution space is also called phenotypic space that is related to the individual fitness. Our contributions are as follows.

About first sub-problem. Similarity guide matrix (SGM) is defined by us to express the population (or subpopulation) distribution in coding space, and two SGMs (i.e. the population SGM S_G and the elitism population SGM S_E) are used as reference points. Diversity subpopulation searches in opposite-direction region (*od*-Region), which is obtained by using S_G and negative perturbation. This behavior of diversity subpopulation in Fig. 7 always tries to increase Hamming distances between themselves and the current population. Conversely, intensification subpopulation searches in forward-direction region (*fd*-Region), which is obtained by using S_E and positive perturbation. This behavior of intensification subpopulation in Fig. 8 always tries to decrease Hamming distances between themselves and the current elitism population. Therefore, the subpopulation (i.e. diversity subpopulation), which always tries to increase Hamming distances between themselves and the current population, is more suitable for injecting randomness.

About second sub-problem. Linear diversity index (LDI) is defined by us to measure the individual density in coding space. According to LDI, we control the search efforts of diversity and intensification subpopulations in *od*-Region and *fd*-Region, respectively. Therefore, by using population diversity measure (i.e. LDI), we decide how much randomness is injected to different behavior-based subpopulations (i.e. diversity and intensification subpopulations).

The rest of the paper is organized as follows. In Sect. 2, we review related works. In Sect. 3, we introduce BEGA. In Sect. 4, we provide experimental results. In Sect. 5, we give our findings.

2 Related works

2.1 Population diversity measures

Burke et al. (2002, 2004) make a detailed survey of population diversity measures, and analyze the correlated relationships among different measures. Črepinšek et al. (2013) categorize population diversity measures into genotype level, phenotype level, composite genotype–phenotype level measures. In essence, high genotype-level diversity of a population maybe not lead to high behavioral diversity, and behavioral diversity is more important than genotype-level diversity (Darwen and Yao 2001).

2.1.1 Widely used diversity measures

Widely used diversity measures are based on the distances of individual coding. This kind of population diversity measures are based on summing (or averaging) Euclidean distances from every individual coding to the center-point, or based on summing (or averaging) Euclidean distances between all pairs of individual coding (Morrison and De Jong 2002; Ursem 2002). Their linearity features are based on the distance's linearity in coding space, and this is also their widely used reason. When using these measures, computation cost should be considered.

2.1.2 Recently proposed diversity measures

Mattiussi et al. (2004) propose a diversity measure that is able to be used for the variable length or highly reorganizable coding. Lacevic et al. propose a measure based on Euclidean minimum spanning trees, a class of power-mean-based measures, and three measures based on discrepancy (Lacevic et al. 2007; Lacevic and Amaldi 2010). McGinley et al. (2011) propose healthy population diversity that expresses the population diversity in solution space. These measures are able to improve the performances of one EA, when their features are consistent with specific problems. These measure definitions are generally based on nonlinear functions. This has an effect on their generalities.

2.2 Population diversity controls

Exploration (i.e. diversity-based search in the entirely new regions) and exploitation (i.e. intensification-based search in the neighborhoods of the previously visited regions) are two cornerstones of problem solving by search (Eiben and Schippers 1998; Črepinšek et al. 2013). De Jong (2007) gives us a historical overview of parameter setting, which includes parameter tuning before the run and parameter control during the run (Eiben et al. 1999). In this paper, parameter-tuning-based, parameter-control-based, and strategy-based controls are reviewed as follows.

2.2.1 Parameter-tuning-based controls

Smit and Eiben discuss three layers (i.e. design, algorithm, and application layers) of parameter tuning (Smit and Eiben 2009; Eiben and Smit 2011). Generally, parameter setting of one operator is easy to be set, because this operator function is single. For instance, the function of each operator is single in simple GA (SGA). Thus, to find a satisfactory parameter setting is easy in SGA. Choosing random number generators (RNGs) is important to the robustness of EAs, and the study between RNGs and the robustness of EAs is a neglected research (Eiben and Smit 2011).

2.2.2 Parameter-control-based controls

CHC (Eshelman 1991) uses convergence threshold to decide whether to reinitialize the population. According to saw-tooth function, saw-tooth GA reinitializes a part of populations (Koumousis and Katsaras 2006). According to the numbers of successful matings and failed matings, variable dissortative mating GA (VDMGA) adjusts convergence threshold (Fernandes and Rosa 2006, 2008). Instead of the fixed crossover probability and mutation probability, zhang et al. (2007) propose the fuzzy-controlled crossover probability and mutation probability. Alba and Dorronsoro (2005) study the influence of the ratio for structured dynamical populations in cellular GA. In dual-population GA (DPGA), according to the distance fitness function, reserve population adjusts the distance from main population that is evaluated by using the fitness function (Park and Ryu 2010). Through crossbreeding, DPGA protects population diversity. According to different parameter-control-based rules, these researches realize population behavior diversities. In addition, population behavior diversities are able to be evaluated by using multi criteria analysis to improve the robustness of EAs.

2.2.3 Strategy-based controls

Multi optimized rules for tournament selection are proposed by Chen et al. (2009), and these rules are characterized by their dynamic properties. The individuals, with the poor fitness and lower contribution to population diversity, are replaced with offspring in contribution of diversity/replace worst strategy (Lozano et al. 2008). Brain storm optimizations (BSOs) are proposed as a new swarm-based algorithm (Shi 2015). Finding the search regions with the promising solutions in BSOs is worthwhile work. But, it is difficult in real coding space. For instance, one dimension in binary coding space only contains two coding types: ‘0’ and ‘1’. However, one dimension in real coding space contains infinitely many real numbers. BSOs are characterized by grouping strategies, such as K-means-clustering, K-medians-clustering, simple, random, and objective-space-based grouping strategies (Zhu and Shi 2015; Zhan et al. 2012; Cao et al. 2015; Shi 2015). In addition, BSOs are inspired by brain storm process, and brain storm process includes the thinking process. In some sense, the thinking process is a highlight of BSOs. Instead of Gaussian distribution, Lee and Yao (2004) propose a mutation based on Levy distribution, and this distribution has an infinite second moment. Thus, this mutation is more likely to generate offspring, which are far away from parents. Scatter search (Glover 1997) explores solution space by using a set of reference individuals to generate offspring, which simultaneously uses diversification and intensification strategies (Resende et al. 2010). Based on this systematic set of reference individuals, offspring generally obtain better performances.

These strategies are important to balance exploration and exploitation in EAs. To find optimal solutions, there are three obstacles: the huge size of coding space, the huge size of solution space, and various fitness-function-mapping relationships (FRs). If one FR is one-to-one linear mapping, finding optimal solutions is easy by using uniformly-spaced sampling method, even for the huge sizes of coding and solution spaces. Of the three obstacles, various FRs are the most difficult. Dynamic varying rules for population diversity strategies are the interesting research to solve various FRs.

2.2.4 Discussions

In essence, population diversity measures indirectly or directly express the population distribution in coding, solution, or composite coding-solution spaces. Population diversity controls are different in many respects, especially for balancing exploration and exploitation.

How to control population diversity by using diversity measures is an interesting problem. Generally, feedback control is a good scheme. According to diversity measures, feedback control schemes do population diversity controls in real time. In feedback control schemes, realizing population behavior diversities is the key to success. Each behavior-based method should only search a single region. Moreover, we should avoid that two or more behavior-based methods search a same region.

3 Balanced-evolution genetic algorithm

In BEGA, S_G and S_E are reference points for diversity and intensification subpopulations, respectively.

3.1 Similarity guide matrix

3.1.1 Motivation

Population similarity researches focus on the similarity of individual coding. Schema is a template that identifies a subset of strings with similarities at certain string positions

(Holland 1975). Probability vector in compact GA expresses coding similarity in binary-coding problems (Harik et al. 1999). As a kind of estimation of distribution algorithms, quantum-inspired evolutionary algorithm (QEA) uses Q-bit to represent the population in binary-coding problems (Han and Kim 2002; Platel et al. 2009).

The motivation of SGM is to express the population distribution in coding space. Different from the above researches, we use SGM as the reference point in coding space. According to different reference points, we define different search regions. Different from probability vector and Q-bit, SGM is able to be suitable for binary-coding and symbol-coding problems, simultaneously.

3.1.2 Definition

SGM is a two-dimensional matrix to express the population (or subpopulation) distribution in coding space. The horizontal and vertical axes of this matrix correspond to coding position (also called gene location, i.e. j in Eq. (1)) and coding type (also called gene type, i.e. i in Eq. (2)), respectively. The definition of SGM is given by

$$S_G = \{ s_{v1} \quad s_{v2} \quad \ldots \quad s_{vj} \quad \ldots \quad s_{vN} \} \quad 1 \le j \le N \tag{1}$$

$$s_{vj} = [s_{0,j} \cdots s_{i,j} \cdots s_{b_j,j}]^T \quad 0 \le i \le b_j \tag{2}$$

$$B = \{ b_1 \quad b_2 \quad \ldots \quad b_j \quad \ldots \quad b_N \} \quad 1 \le b_j \le b_{\max} \tag{3}$$

$$X = \{ x_1 \quad x_2 \quad \ldots \quad x_j \quad \ldots \quad x_N \} \tag{4}$$

$$s_{i,j} = p_{i,j}/P \quad 0 \le i \le b_j \tag{5}$$

$$\forall j \quad \sum_{i=0}^{b_j} s_{i,j} = 1 \quad 0 \le i \le b_j \tag{6}$$

where N is coding length, and s_{vj} is similarity guide vector (SGV). b_j is the upper bound of each genome x_j, and $b_{\max}$ is the upper bound of each b_j ($1 \le b_j \le b_{\max}$). In symbol-coding problems, $b_{\max}$ is greater than 1, and b_j may be any integer from 1 to $b_{\max}$. In binary-coding problems, $b_{\max}$ and b_j are always equal to 1. X is individual coding, and x_j is the value of each genome ($0 \le x_j \le b_j$). $p_{i,j}$ is the number of individuals with i's at the jth position, and P is population size.

In Fig. 1, a symbol-coding example of SGM is given to demonstrate the relationships among coding position, coding type, individuals, and SGM. In addition, a binary-coding example of SGM is given in Fig. 2. In a global view, Fig. 2 demonstrates the relationship between coding space and its uniform distribution. Conversely, in a local view, Fig. 2 explains the relationship between individuals and their SGM. By comparing this global view with this local view, SGM is suitable for using as a reference point to express the population distribution in coding space.

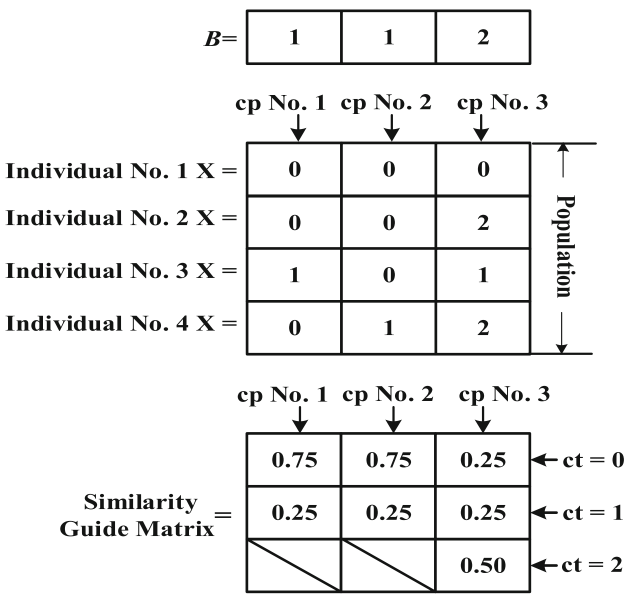


Fig. 1 A symbol-coding example to demonstrate the relationships among coding position, coding type, individuals, and similarity guide matrix ($N = 3$, $P = 4$, $b_{\max} = 2$)

3.1.3 Principle analysis

The principle of SGM is analyzed as follows. First, SGM indicates the search directions of the promising solutions (i.e. solutions with development potential), especially for S_E. In unimodal problems, the promising solutions may be near the local/global optimal solutions. In multimodal problems, the promising solutions may be near the local/-global optimal solutions, or may be development-potential solutions that are able to make populations escape from the local optimal solutions. Secondly, SGMs are used as reference points to define different search regions. This is feasible. In coding space, we only know individual coding. By comparing with the huge size of coding space, these information is very little. But, based on SGMs in Eq. (1), this kind of reference point is sensitive to reflect the coding change of each individual. Therefore, by using this kind of reference point, the search regions of different subpopulations are able to be defined.

To further demonstrate the principle of SGM, a unimodal problem case is analyzed as follows. Figure 3a shows the relationships among coding position, coding type, coding space, its uniform distribution, individual coding, fitness, and the global optimal solution. With generations increasing (from Fig. 3b–c), SGM becomes nonuniform. In Fig. 3c, each SGV gives the value judgment at each position of the promising solution. In addition, these indicate that SGM is sensitive to the coding change of each individual.

The process of breeding new individuals, based on SGM, is given in Algorithm 1.

Algorithm 1 Breed new individuals, based on SGM

```
1:  Step 1: Compute similarity threshold matrix T of SGM by using
2:      $t_{i,j} = s_{0,j}$                  $i = 0$          $1 \le j \le N$   $0 \le i \le b_j$   (7)
3:      $t_{i,j} = t_{i-1,j} + s_{i,j}$      $i \ne 0$
4:      where $t_{i,j}$ is an element at the $(i,j)$ position of similarity threshold matrix $T$.
5:  Step 2: Breed a new individual by using
6:      $x_j = 0$     $r_j \le t_{0,j}$                      $1 \le j \le N$   $0 \le i \le b_j$   (8)
7:      $x_j = i$     $t_{i-1,j} < r_j \le t_{i,j}$
8:      where $x_j$ is the $j$th element of the new individual. $r_j$ is a random number uniformly distributed in (0, 1).
```

3.2 Linear diversity index

3.2.1 Motivation

In biology, biological diversity measures are important to understand the degree of species richness, such as Jaccard Index (Jaccard 1912). Diversity-related concepts in biology are also suitable for EAs. We make some modifications to Jaccard Index for its applications in EAs. These modifications mainly focus on the differences between biology and EAs. Inspired by Jaccard Index, we propose LDI. The motivation of LDI is to measure the individual density in coding space. From a coding-space perspective, LDI is able to be used for population diversity control.

3.2.2 Definition

First, we define the center-point individual X_{cp} of the population by

$$X_{cp} = \{IndexMax(s_{v1}), \ldots IndexMax(s_{vj}), \ldots IndexMax(s_{vN})\} \quad (9)$$

where *IndexMax*() returns the index of the maximum value in SGV. X_{cp} includes the index of the maximum value in

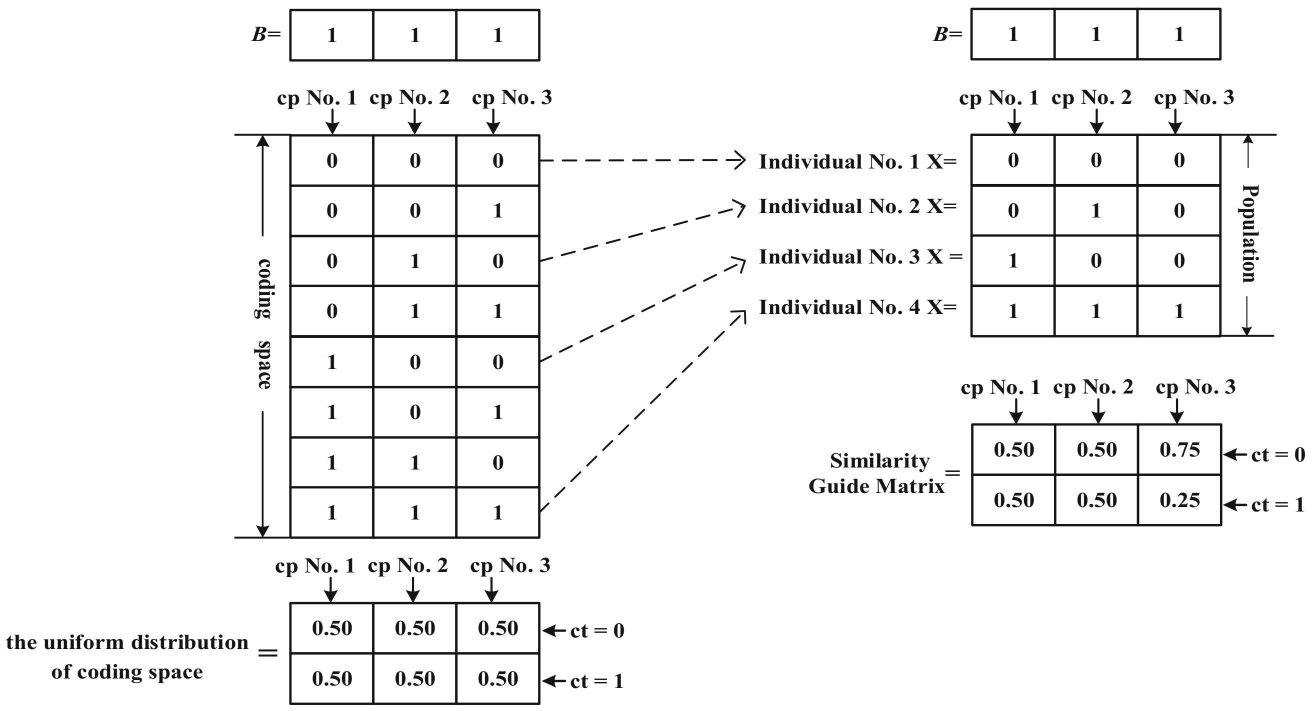
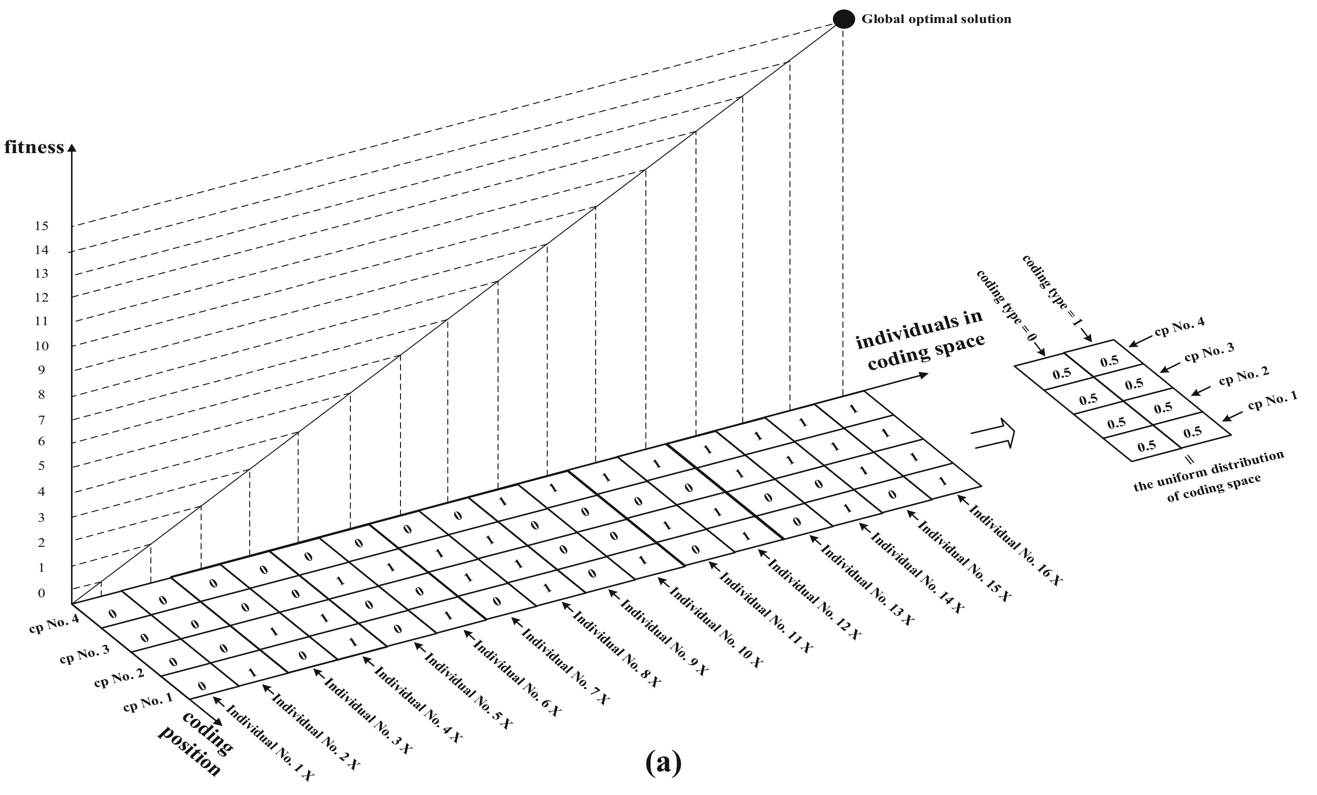


Fig. 2 A binary-coding example to demonstrate that similarity guide matrix is suitable for using as a reference point to express the population distribution in coding space ($N = 3$, $P = 4$, $b_{\max} = 1$)



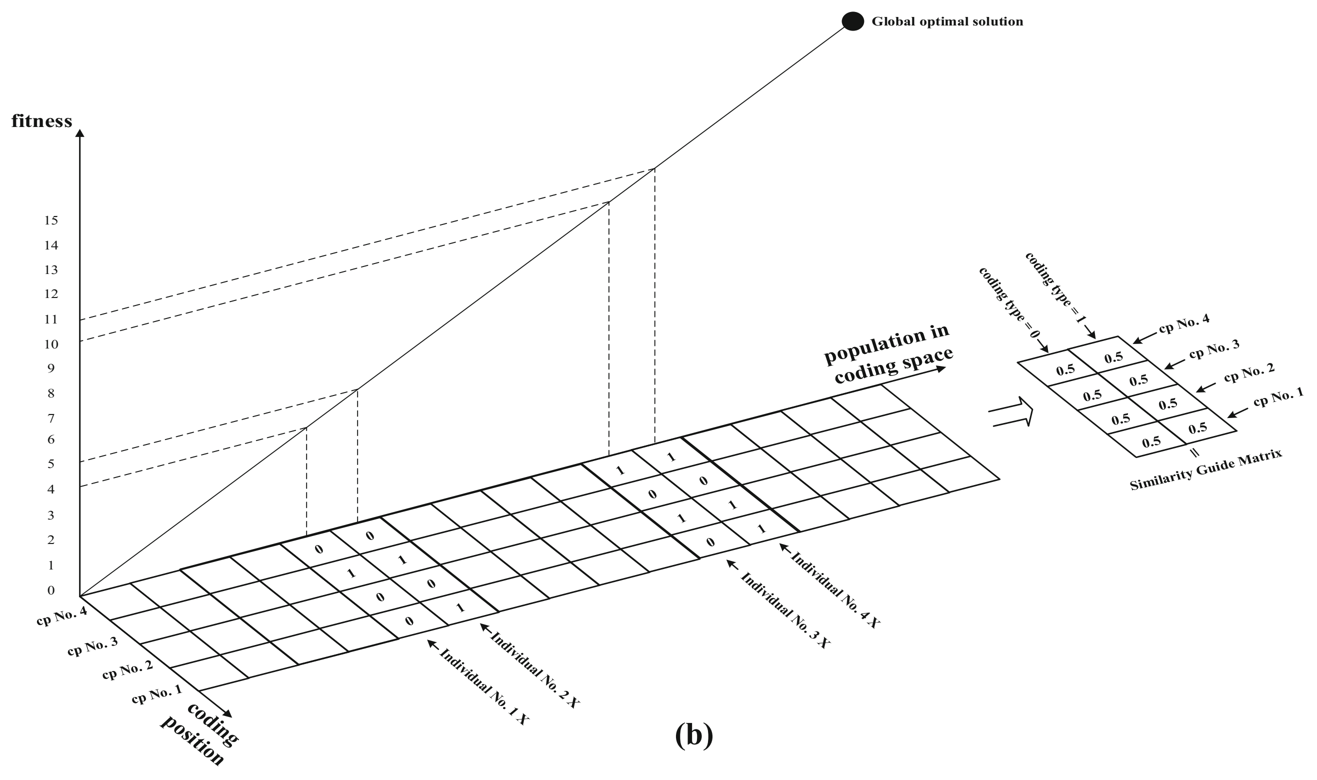
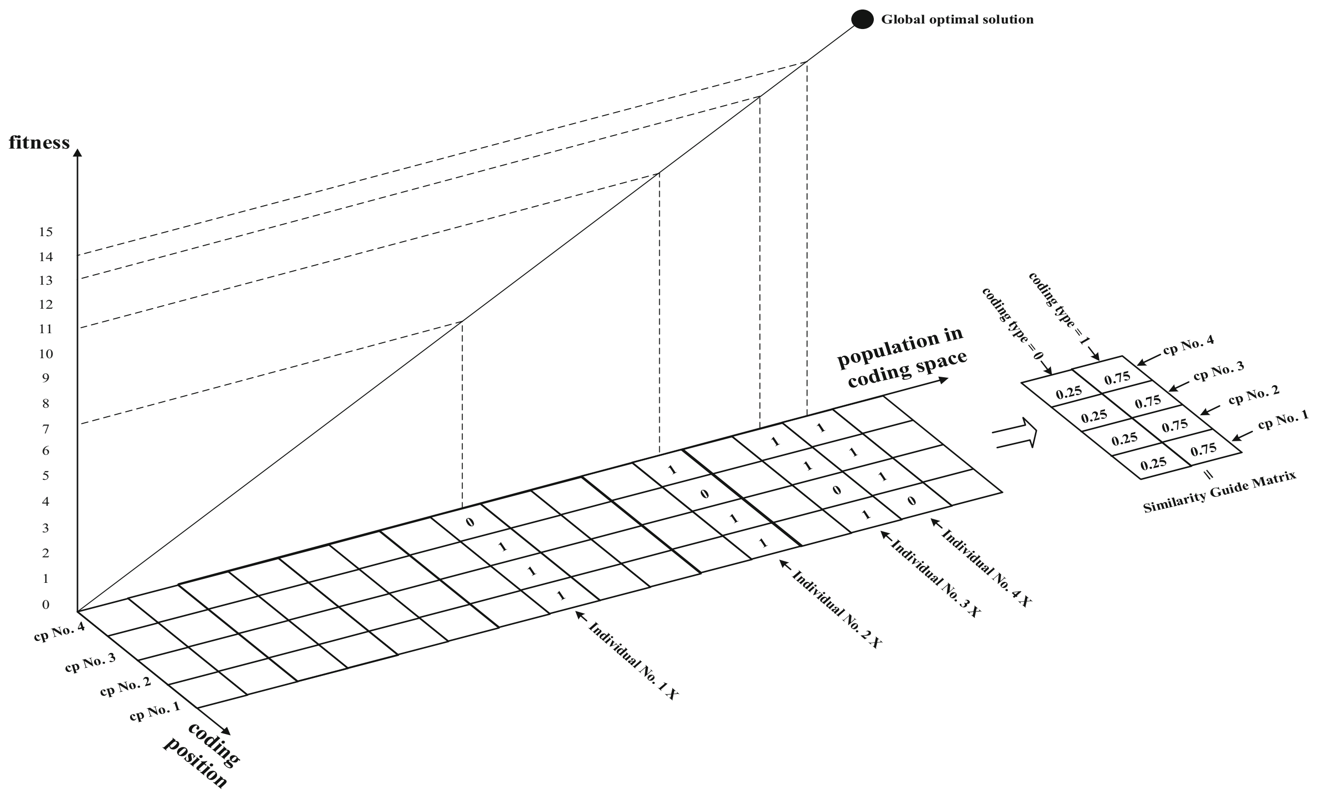


Fig. 3 A unimodal problem case analysis to demonstrate the principle of similarity guide matrix. **a** Three-dimensional graph about coding position, fitness, and individuals in coding space. **b** The population distribution and similarity guide matrix in the early-evolution generations. **c** The population distribution and similarity guide matrix in the later-evolution generations



(c)

Fig. 3 continued

each SGV. The reason is that the index of the maximum value in each SGV is equal to the majority value for all of coding values in the corresponding coding position. For instance, the index of the maximum value in SGV S_{v1} of Fig. 1 is equal to 0, and the majority value in coding position No. 1 of Fig. 1 is also equal to 0.

LDI D_l expresses the individual density around X_{cp} in coding space, which is defined by

$$D_l = \sum_{p=1}^{P} D_p / P \quad 1 \le p \le P \tag{10}$$

$$D_p = Hamdis(X_p, X_{cp})/N \tag{11}$$

where P is population size, N is coding length, and *Hamdis*() returns Hamming distance between X_{cp} and the pth individual X_p.

3.2.3 Principle analysis

The principle of LDI is analyzed as follows. First, X_{cp} always exists in coding space, and does not always exist in solution space. In Eq. (9), X_{cp} may be regarded as individual coding. However, this individual may not be a feasible solution in the solution space of constrained optimization problems. The reason is that this X_{cp} is not satisfied with constraints, and we cannot obtain the fitness value of X_{cp}. However, X_{cp} always exists in coding space, because coding space is not related with constraints and fitness. Secondly, in coding space, linearity is an important condition to control population diversity. In Eq. (9), the approximation between X_{cp} and S_G is acceptable. From a coding-position estimation perspective, each element in X_{cp} is the majority value in each coding position. Using the majority value as the center point in each coding position is suitable for the center-point meaning to measure diversity. In addition, Hamming distance between two individuals is linear, and LDI is also linear. Thus, LDI is suitable for population diversity control.

To further demonstrate the principle of LDI, we give a case analysis. In Fig. 4a, individual-loose distribution means long Hamming distances between X_{cp} and each individual, and indicates good diversity. In Fig. 4b, individual-close-together distribution means short Hamming distances between X_{cp} and each individual, and indicates poor diversity. With generations increasing, X_{cp} changes continually. Meanwhile, population diversity gradually decreases from individual-loose distribution to individual-close-together distribution.

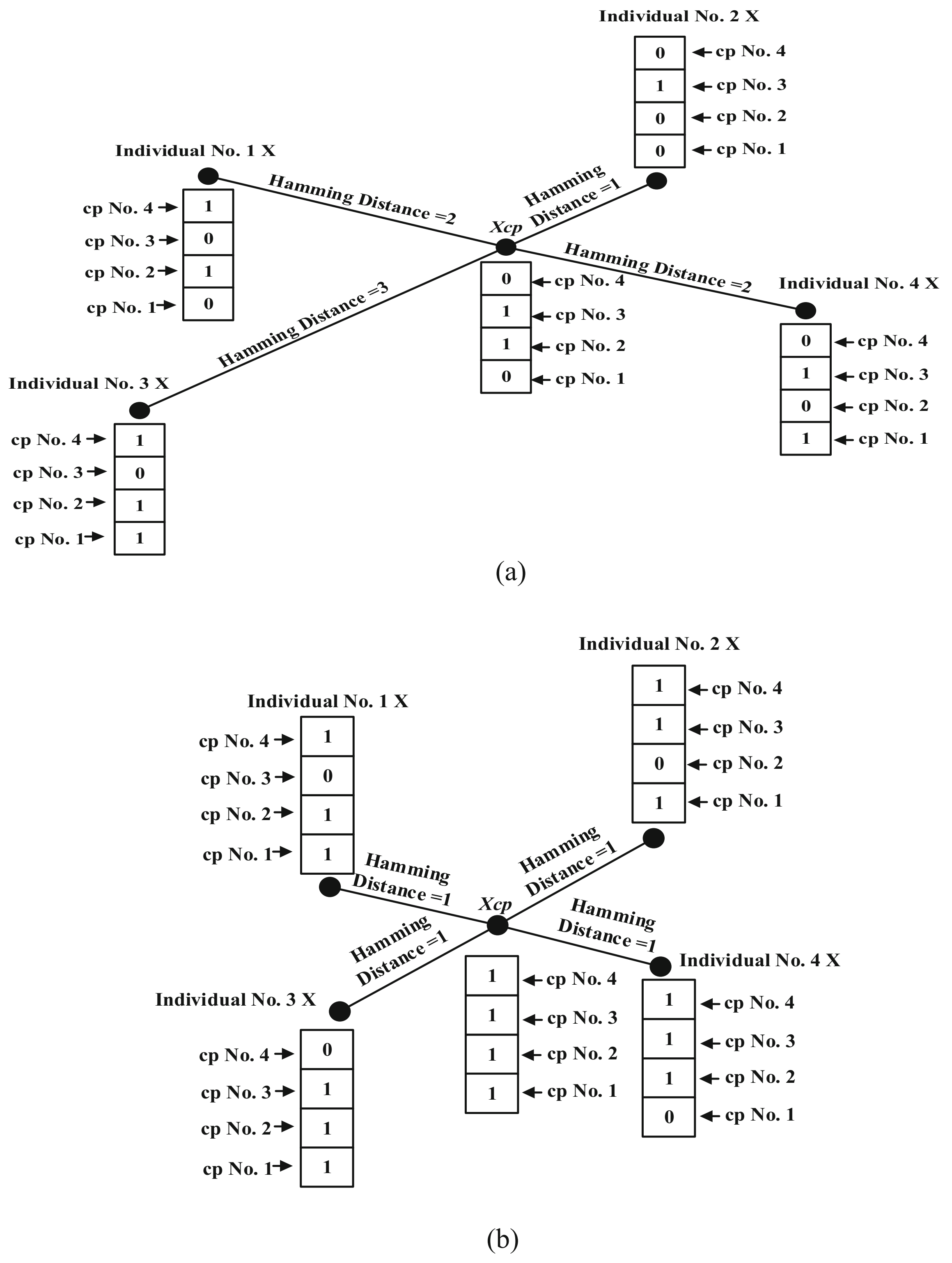


Fig. 4 A case analysis of linear diversity index to demonstrate the individual density around the center-point individual X_{cp}. **a** Individual-loose distribution. **b** Individual-close-together distribution

3.3 Algorithm design

According to S_G and S_E as reference points, BEGA use negative and positive perturbations to control the search directions of diversity and intensification subpopulations, respectively. In addition, we explain the balanced-evolution strategy in Sects. 3.3.2 and 3.3.3.

3.3.1 Framework

The procedure of BEGA is given in Algorithm 2. Balanced-evolution procedure in Fig. 5 corresponds to **Step 3–Step 7** in Algorithm 2.

We define the first and second stages by using LDI D_l, control amplitude CA, and shift limit of population diversity D_{sl}. The first stage is the generations for LDI $D_l > D_{sl}$, and the second stage is the generations for LDI $D_l \leq D_{sl}$. In Fig. 6a, LDI D_l decreases gradually in BEGA, because this is a general law of population diversity in EAs. In addition, CA is used to control the amplitudes of negative and positive perturbations in diversity and intensification methods, respectively. In Fig. 6b, CA is equal to LDI D_l in the first stage, and is equal to D_{sl} in the second stage.

Intensification and diversity subpopulations after updating are 50% of the best individuals in the population. In addition, the procedure of updating elitism population is as follows. In the first stage, by using elite selection, elitism population after updating (EPAU) is obtained for elitism population and intensification subpopulation after updating. In the second stage, EPAU is also obtained by using elite selection. Moreover, Hamming distance between two fitness-adjacent individuals in EPAU is required to be greater than the minimal distance m_d. If the size of the chosen individuals is less than the size of elitism population P_e, we use the best fitness individuals as other individuals in EPAU.

Algorithm 2 Balanced-evolution genetic algorithm

```
1:  Input: population size $P$, coding length $N$, shift limit of population diversity $D_{sl}$, the size of elitism population $P_e$
2:  Step 1: Generate the uniform-distribution initial population, and evaluate this population.
3:  Step 2: According to the initial population, initialize diversity and intensification subpopulations.
4:  Step 3: Compute $S_G$ and LDI $D_l$ of the population.
5:  Step 4: Carry out elite selection to update diversity and intensification subpopulations.
6:  Step 5: Compute control amplitude $CA$, and update the elitism population.
7:  Step 6: Carry out diversity method, and evaluate the new diversity subpopulation.
8:  Step 7: Carry out intensification method, and evaluate the new intensification subpopulation.
9:  Step 8: Check whether termination condition is satisfied or not. If it is not satisfied, go to Step 3.
10: Step 9: The individual with the best fitness represents the final solution.
```

3.3.2 Diversity method

Algorithm 3 Diversity method

```
1:  Step 1: According to $S_G$, compute each vector of $np$-SGM $M_N$ as follows.
2:          if $rand()$<control amplitude $CA$
3:              Compute negative perturbation $r_n$, and inject negative perturbation $r_n$ by using
4:                  $r_n = CA \times rand() \times Max(s_{vj})$                                    (12)
5:                  $m_{k,j} = s_{k,j} - r_n$                                                        (13)
6:                  $m_{l,j} = s_{l,j} + r_n$                                                        (14)
7:              Except for $m_{k,j}$ and $m_{l,j}$, other elements in the $j$th vector of $M_N$ are equal to the corresponding elements
8:              in the $j$th SGV $s_{vj}$ of $S_G$. $rand()$ generates a random number uniformly distributed in (0, 1), and $Max()$
9:              computes the maximum of a vector. $k$ is the index of the maximal element of $s_{vj}$ in $S_G$. $l$ is a randomly
10:             selected index ($k \neq l$).
11:         else
12:             The $j$th vector of $M_N$ is equal to the $j$th vector of $S_G$.
13: Step 2: Breed temporary individuals by using $np$-SGM $M_N$ and Algorithm 1, and execute crossover by using
14:             $Q = \{q_1 \quad q_2 \quad \ldots \quad q_j \quad \ldots \quad q_N\}$                  (15)
15:             $o_j = \begin{cases} q_j & 0 \le r_j \le 0.5 \\ x_j & 0.5 < r_j \le 1 \end{cases}$      (16)
16:         where $Q$ is a temporary individual, and $r_j$ is a random number uniformly distributed in (0, 1). $x_j$ is the
17:         value at the $j$th position of a decision variable $X$ (corresponding to diversity subpopulation after updating),
18:         and $o_j$ is the value at the $j$th position of a new individual $O$.
19: Step 3: Execute mutation according to mutation probability $p_m$ in Eq. (17), and $m_s$ is multiplier factor.
20:             $p_m = (1.0 + m_s \times CA)/N$                                                    (17)
```

od-Region is determined by M_N. In Fig. 7a, negative perturbation changes the coding distribution of temporary individuals. In Fig. 7b, comparing with Hamming distances between diversity subpopulation and diversity subpopulation, Hamming distances between temporary individuals and diversity subpopulation increase. Moreover, the new diversity subpopulation is obtained by using temporary individuals and diversity subpopulation. Therefore, Hamming distances between the new diversity subpopulation and diversity subpopulation also increase. This realizes the exploration of the new diversity subpopulation, and improves the robustness of BEGA.

To explain the behavior features of diversity subpopulation about the balanced-evolution strategy, we use the examples in Figs. 6 and 7. The behavior features of diversity subpopulation are as follows. CA is important to

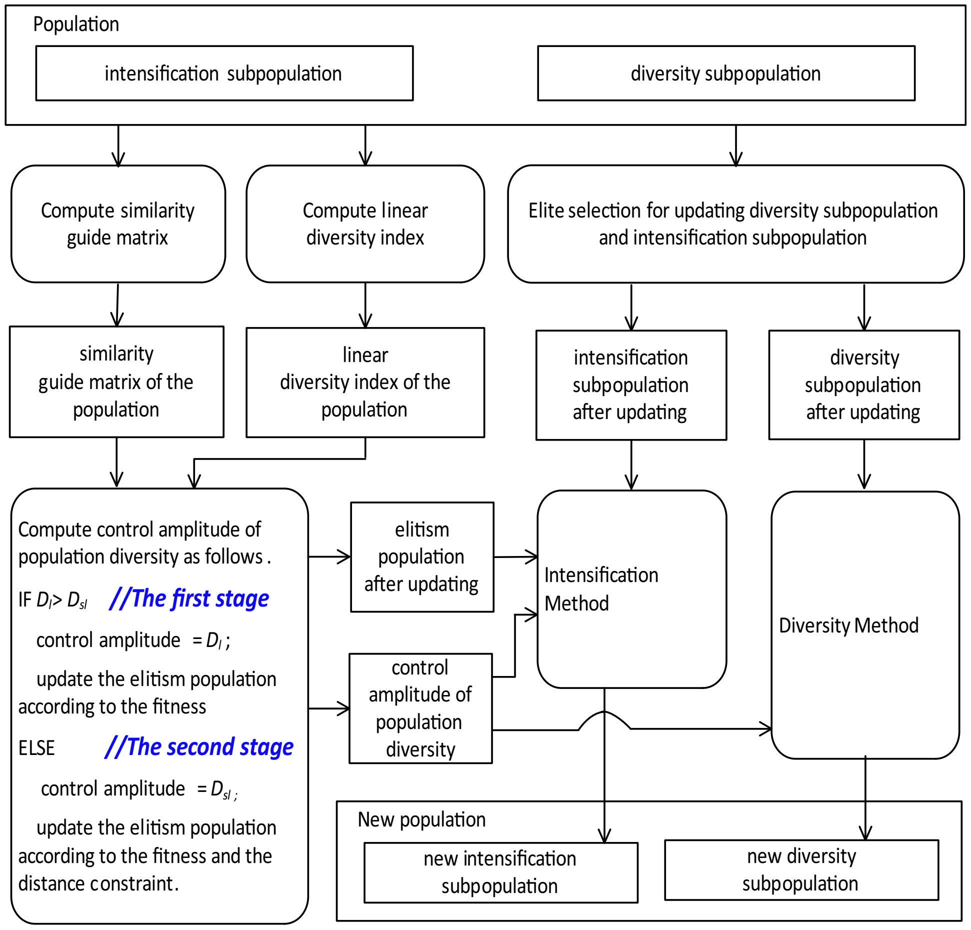

Population
intensification subpopulation
diversity subpopulation
Compute similarity guide matrix
Compute linear diversity index
Elite selection for updating diversity subpopulation and intensification subpopulation
similarity guide matrix of the population
linear diversity index of the population
intensification subpopulation after updating
diversity subpopulation after updating
Compute control amplitude of population diversity as follows .
IF D_l> D_{sl} //The first stage
control amplitude = D_l ;
update the elitism population according to the fitness
ELSE //The second stage
control amplitude = D_{sl} ;
update the elitism population according to the fitness and the distance constraint.
elitism population after updating
control amplitude of population diversity
Intensification Method
Diversity Method
New population
new intensification subpopulation
new diversity subpopulation

Fig. 5 The balanced-evolution procedure of the population in each generation

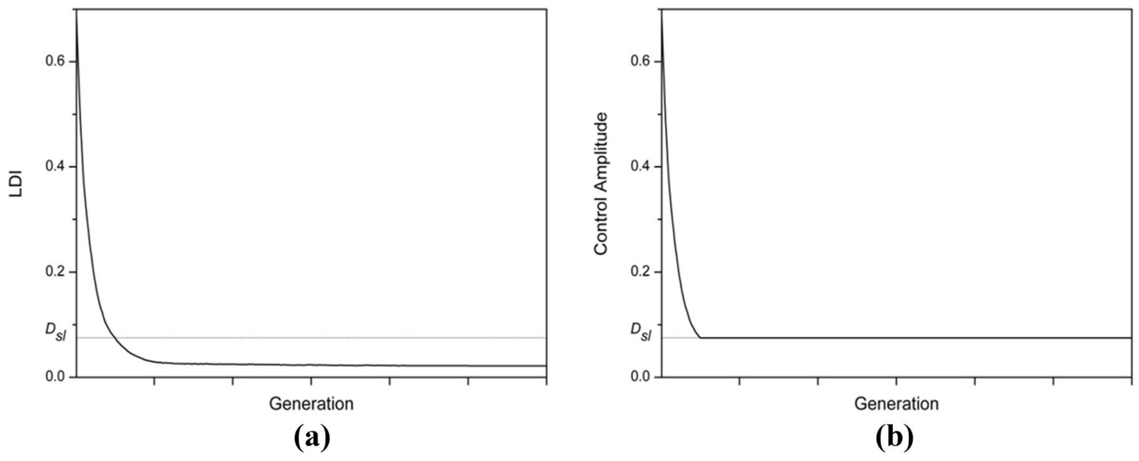


Fig. 6 An example of linear diversity index D_l, control amplitude *CA*, and shift limit of population diversity D_{sl} in the first and second stages

control the amplitude of negative perturbation. In Fig. 6b, with LDI D_l decreasing, *CA* is decreasing in the first stage, and then is equal to D_{sl} in the second stage. This kind of *CA* setting is suitable for improving the exploration abilities of diversity subpopulation, when the population diversity is decreasing in Fig. 6a. In Fig. 7, *CA* setting is also suitable for increasing Hamming distances between temporary individuals and diversity subpopulation.

3.3.3 Intensification method

Algorithm 4 Intensification method

```
1:  Step 1: Compute $S_E$, and copy the best individual into the next generation.
2:  Step 2: According to $S_E$, compute each vector of pp-SGM $M_P$ as follows.
3:          Compute positive perturbation $r_p$, and inject positive perturbation $r_p$ by using
4:          $r_p = CA \times rand() \times Min(s_{vj})$          (18)
5:          $m_{k,j} = s_{k,j} - r_p$          (19)
6:          $m_{l,j} = s_{l,j} + r_p$          (20)
7:          Except for $m_{k,j}$ and $m_{l,j}$, other elements in the jth vector of $M_P$ are equal to the corresponding elements in
8:          the jth SGV $s_{vj}$ of $S_E$. rand() generates a random number uniformly distributed in (0, 1), and Min()
9:          computes the minimum of a vector. k is the index of the minimal element of $s_{vj}$ in $S_E$. l is a randomly
10         selected index ($k \neq l$).
11: Step 3: Breed temporary individuals by using pp-SGM $M_P$ and Algorithm 1. Similar to Algorithm 3, execute
12:         crossover between temporary individuals and intensification subpopulation after updating.
13: Step 4: Execute mutation according to mutation probability $p_m$ ($p_m = 1/N$).
```

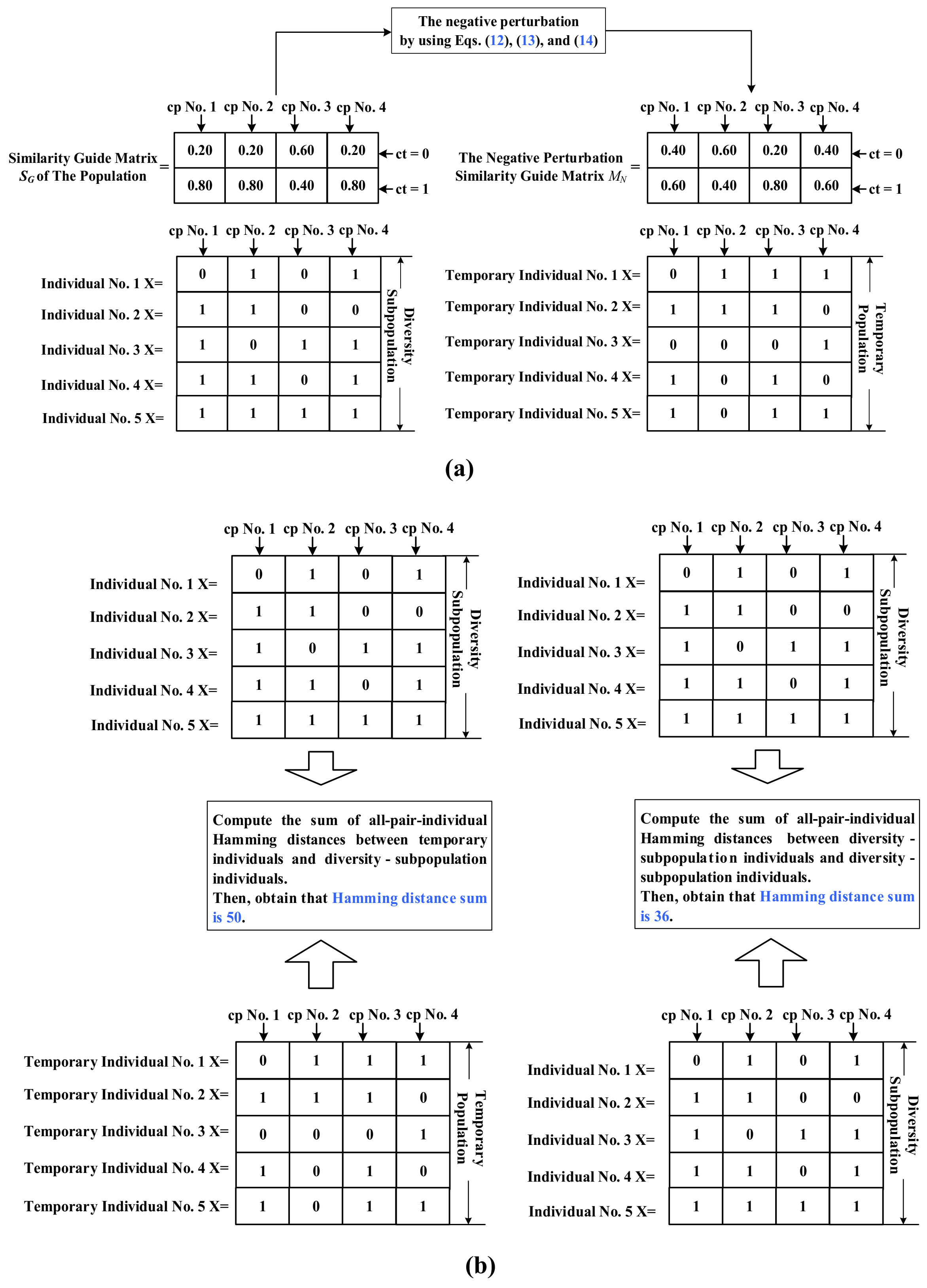
fd-Region is determined by M_P. Similarly to Fig. 7, an example of intensification method is given in Fig. 8. Positive perturbation is the reason that Hamming distances between the new intensification subpopulation and elitism population decrease. This realizes the exploitation of the new intensification subpopulation, and improves the efficiency of BEGA.

To explain the behavior features of intensification subpopulation about the balanced-evolution strategy, we use the examples in Figs. 6 and 8. The behavior features of intensification subpopulation are as follows. *CA* is important to control the amplitude of positive perturbation. In Fig. 6b, with LDI D_l decreasing, *CA* is decreasing in the first stage, and then is equal to D_{sl} in the second stage. This kind of *CA* setting is suitable for improving the exploitation abilities of intensification subpopulation, when the population diversity is decreasing in Fig. 6a. In Fig. 8, *CA* setting in Fig. 6a is also suitable for reducing Hamming distance between temporary individuals and elitism population.

3.4 Discussions

3.4.1 Cooperation

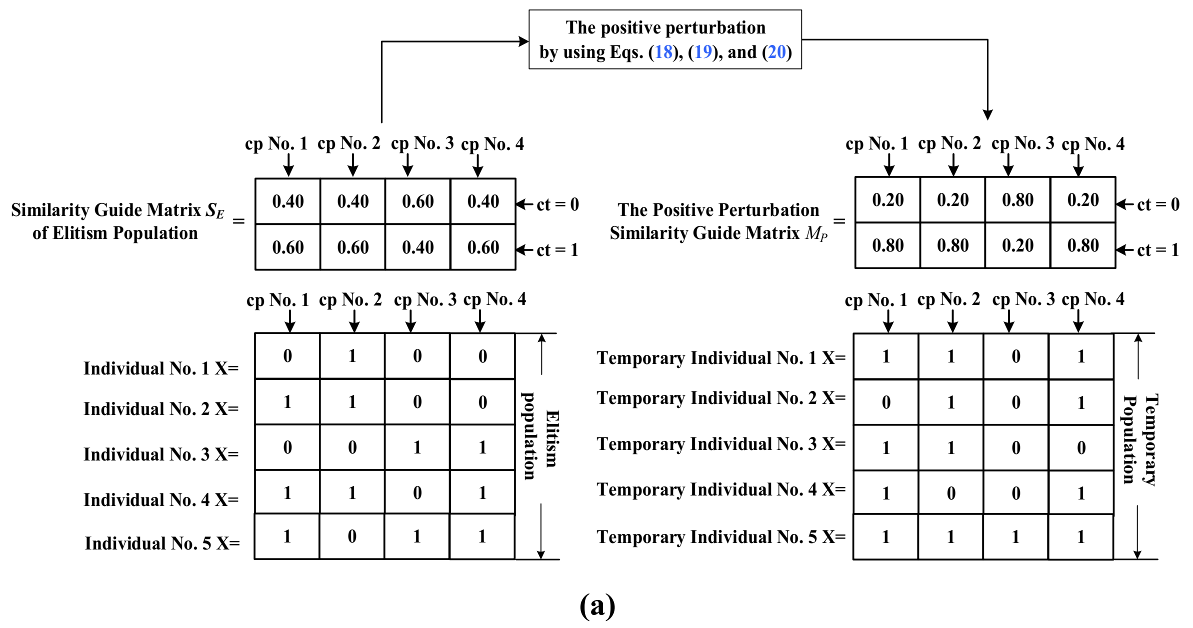
The relationship between diversity subpopulation and intensification subpopulation is a kind of cooperation. First, the definitions of *od*-Region and *fd*-Region are the foundation of the cooperation. To search *od*-Region and *fd*-Region, diversity and intensification subpopulations cooperate with each other. The diversity-subpopulation size is equal to the intensification-subpopulation size. Thus, their search efforts are the same. Secondly, another kind of cooperation is the individual exchanges between diversity subpopulation and intensification subpopulation. In the second stage, there are more and more individual exchanges, which should be called individual shuffling. This kind of individual shuffling is important to find the new promising solutions.


The negative perturbation by using Eqs. (12), (13), and (14)
cp No. 1 cp No. 2 cp No. 3 cp No. 4
Similarity Guide Matrix S_G of The Population =
0.20 0.20 0.60 0.20 ← ct = 0
0.80 0.80 0.40 0.80 ← ct = 1
The Negative Perturbation Similarity Guide Matrix M_N =
0.40 0.60 0.20 0.40 ← ct = 0
0.60 0.40 0.80 0.60 ← ct = 1
Individual No. 1 X= 0 1 0 1
Individual No. 2 X= 1 1 0 0
Individual No. 3 X= 1 0 1 1
Individual No. 4 X= 1 1 0 1
Individual No. 5 X= 1 1 1 1
Diversity Subpopulation
Temporary Individual No. 1 X= 0 1 1 1
Temporary Individual No. 2 X= 1 1 1 0
Temporary Individual No. 3 X= 0 0 0 1
Temporary Individual No. 4 X= 1 0 1 0
Temporary Individual No. 5 X= 1 0 1 1
Temporary Population
(a)
(b)

Compute the sum of all-pair-individual Hamming distances between temporary individuals and diversity - subpopulation individuals.
Then, obtain that Hamming distance sum is 50.

Compute the sum of all-pair-individual Hamming distances between diversity - subpopulation individuals and diversity - subpopulation individuals.
Then, obtain that Hamming distance sum is 36.

Fig. 7 An example to demonstrate that Hamming distances between temporary individuals and diversity subpopulation increase ($N = 4$, $b_{\max} = 1$). **a** The relationship between negative perturbation and temporary individuals. **b** The sum of all-pair-individual Hamming distances between temporary individuals and diversity-subpopulation individuals is 50. Conversely, the sum of all-pair-individual Hamming distances between diversity-subpopulation individuals and diversity-subpopulation individuals is 36


The positive perturbation by using Eqs. (18), (19), and (20)
Similarity Guide Matrix S_E of Elitism Population =
cp No. 1 cp No. 2 cp No. 3 cp No. 4
0.40 0.40 0.60 0.40 ct = 0
0.60 0.60 0.40 0.60 ct = 1
The Positive Perturbation Similarity Guide Matrix M_P =
0.20 0.20 0.80 0.20 ct = 0
0.80 0.80 0.20 0.80 ct = 1
Individual No. 1 X= 0 1 0 0
Individual No. 2 X= 1 1 0 0
Individual No. 3 X= 0 0 1 1
Individual No. 4 X= 1 1 0 1
Individual No. 5 X= 1 0 1 1
Elitism population
Temporary Individual No. 1 X= 1 1 0 1
Temporary Individual No. 2 X= 0 1 0 1
Temporary Individual No. 3 X= 1 1 0 0
Temporary Individual No. 4 X= 1 0 0 1
Temporary Individual No. 5 X= 1 1 1 1
Temporary Population
(a)

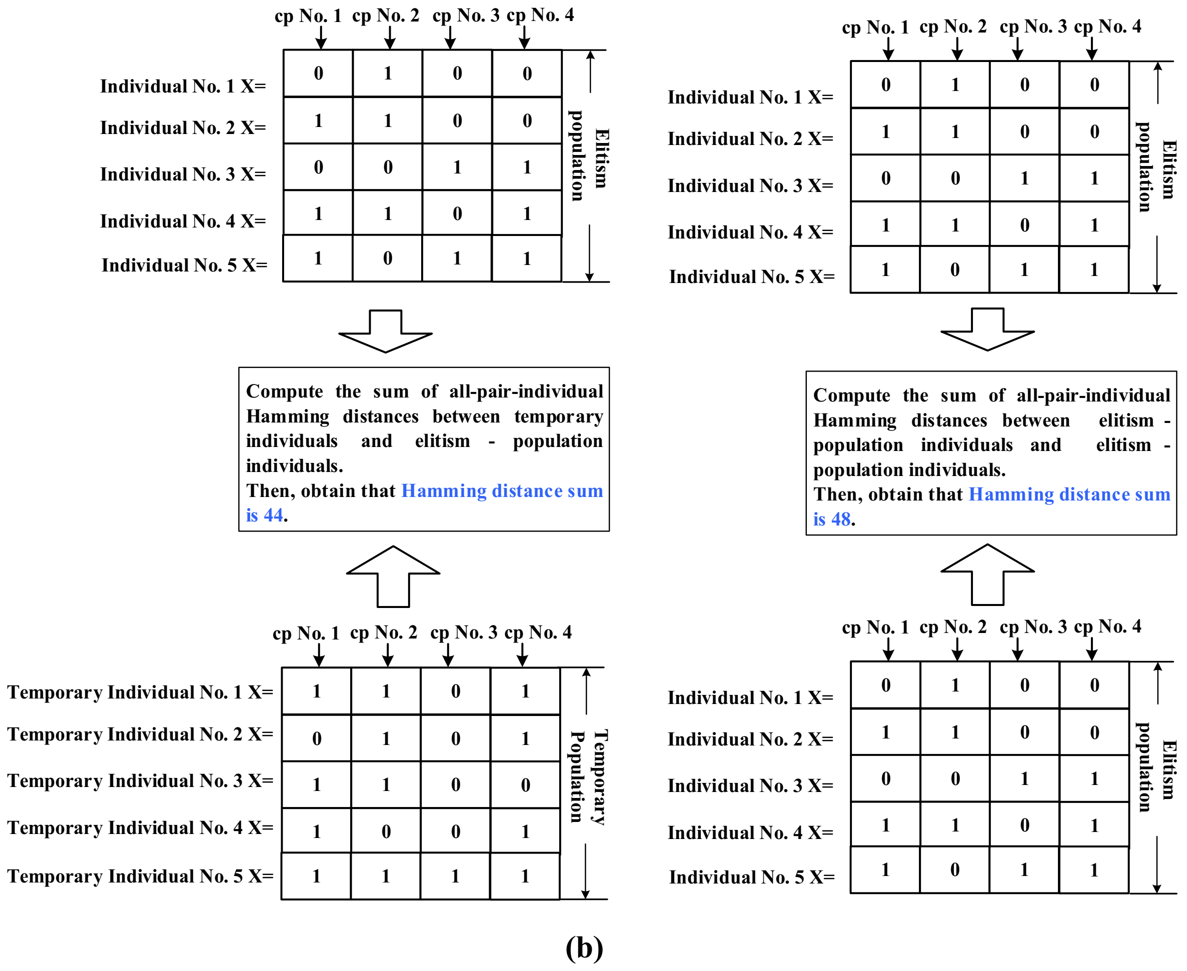

Compute the sum of all-pair-individual Hamming distances between temporary individuals and elitism - population individuals.
Then, obtain that Hamming distance sum is 44.
Compute the sum of all-pair-individual Hamming distances between elitism - population individuals and elitism - population individuals.
Then, obtain that Hamming distance sum is 48.
Individual No. 1 X= 0 1 0 0
Individual No. 2 X= 1 1 0 0
Individual No. 3 X= 0 0 1 1
Individual No. 4 X= 1 1 0 1
Individual No. 5 X= 1 0 1 1
Elitism population
Temporary Individual No. 1 X= 1 1 0 1
Temporary Individual No. 2 X= 0 1 0 1
Temporary Individual No. 3 X= 1 1 0 0
Temporary Individual No. 4 X= 1 0 0 1
Temporary Individual No. 5 X= 1 1 1 1
Temporary Population
(b)

Fig. 8 An example to demonstrate that Hamming distances between temporary individuals and elitism population individuals decrease ($N = 4$, $b_{\max} = 1$). **a** The relationship between positive perturbation and temporary individuals. **b** The sum of all-pair-individual Hamming distances between temporary individuals and elitism-population individuals is 44. Conversely, the sum of all-pair-individual Hamming distances between elitism-population individuals and elitism-population individuals is 48

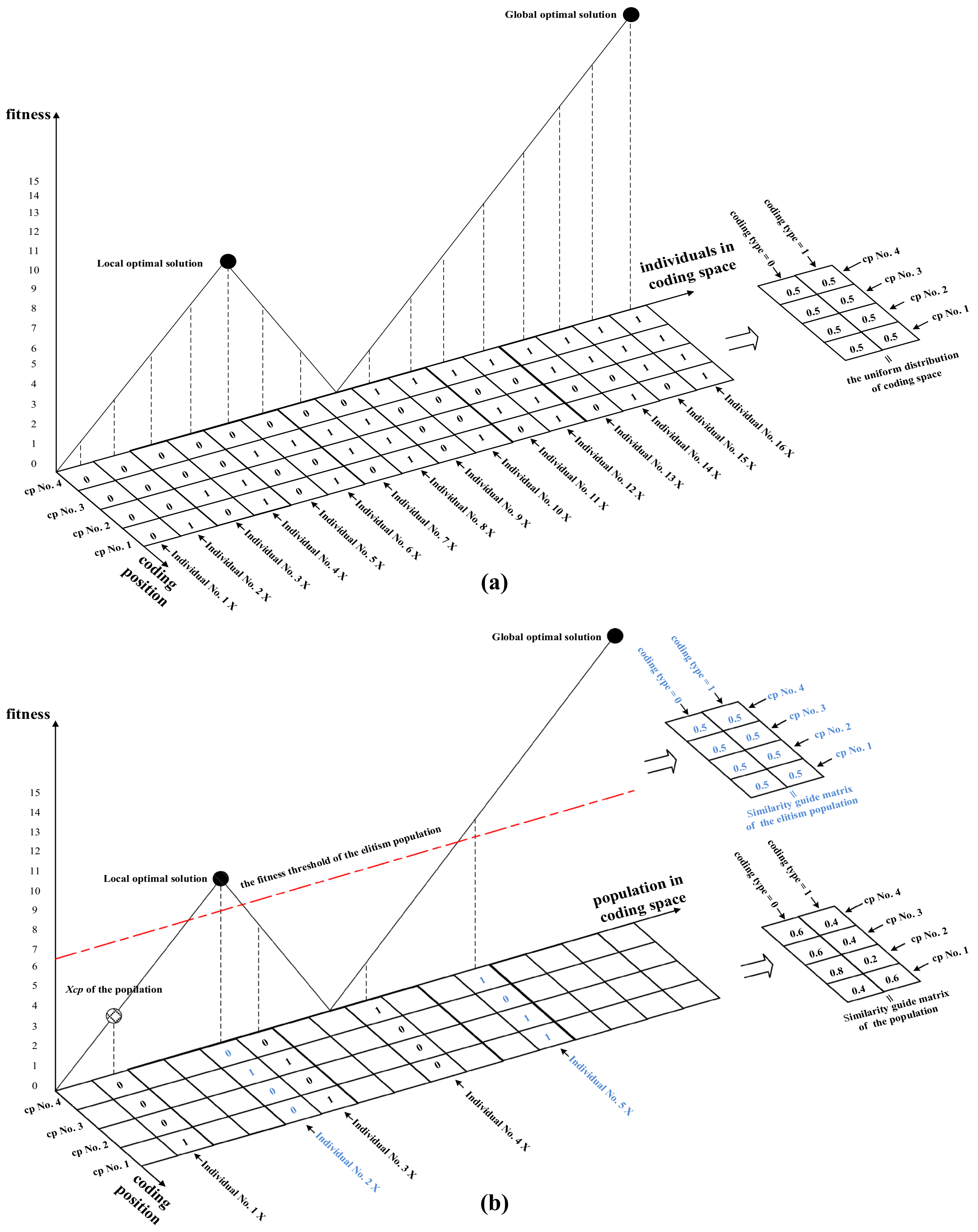
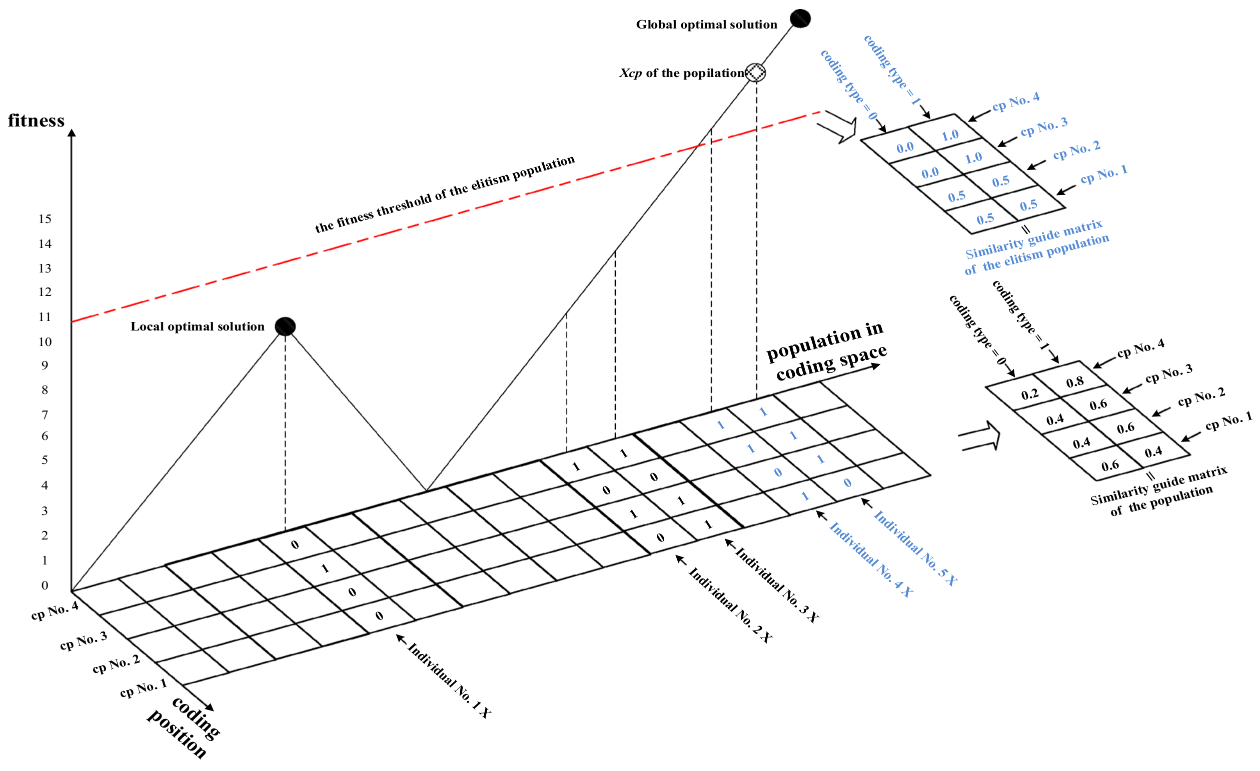


Fig. 9 A multimodal problem example to demonstrate how population escapes from the local optimal solution by using similarity guide matrices of the population and the elitism population. **a** Three-dimensional graph about coding position, fitness, and individuals in coding space. **b** Most of individuals in the population fall into the local optimal solution. **c** Population gradually escapes from the local optimal solution by using similarity guide matrix of the elitism population



(c)

Fig. 9 continued

3.4.2 Efficiency

The no-free-lunch-theorem is always there. Although SGM and LDI are useful, their computation costs have an effect on the efficiency of BEGA. Thus, each kind of computation must promote the population evolution. To realize this goal, we pay more attention to selecting the reasonable search region of each individual. Details are as follows. First, we use S_E in intensification method, and use positive perturbation to reduce Hamming distances between temporary individuals and elitism population. This method is effective to improve the convergence speed of BEGA, especially in the first stage. Secondly, we use S_G in diversity method, and use negative perturbation to increase Hamming distances between temporary individuals and diversity subpopulation. In addition, we use the multiplier factor of mutation operator m_s to scatter diversity subpopulation. These methods of scattering diversity subpopulation are good to explore much more search regions in the first stage. Thus, this strategy is able to improve the convergence speed of BEGA, especially in the second stage.

3.4.3 Adaptability

To avoid falling into the local optimal solution, we use S_G and S_E. The multimodal problems lead to the differences between S_G and S_E. These differences are useful for making the population escape from the local optimal solution. To demonstrate the abilities of escaping from the local optimal solution, a multimodal problem example is given in Fig. 9. First, Fig. 9a presents a multimodal problem about the relationships among coding position, fitness, and individuals in coding space. Secondly, in Fig. 9b, most of individuals in the population (i.e. individual No. 1–No. 4) are close to the local optimal solution, and fall into the local optimal solution. However, two SGMs are different in Fig. 9b. S_G is based on the population, which includes individual No. 1–No. 5. S_E is based on the elitism population, which includes individual No. 2 and No. 5. These differences between S_G and S_E are useful for escaping from the local optimal solution. Thirdly, with the population fitness increasing in Fig. 9c, the fitness threshold of the elitism population also increases. Therefore, S_E gradually changes the population distribution from Fig. 9b to Fig. 9c. Thus, the population escapes from the local optimal solution.

4 Experiments and discussions

4.1 Experimental scheme

4.1.1 Benchmarks

We used 12 combinatorial optimization problems as benchmarks in Appendix 1.

4.1.2 Compared algorithms

We used CHC, DPGA, VDMGA, and QEA as compared algorithms for binary-coding problems, and used greedy GA (GGA) as the compared algorithm for symbol-coding problems in Appendix 2.

4.1.3 Used parameter settings

See Table 1.

4.2 Experiments on convergence

To demonstrate the effectiveness of BEGA, the effectiveness results were given in Table 2. BEGA were acceptable. In Fig. 10, we provided the mean of the best-individual-fitness ranks in binary-coding problems. This indicates that the convergence performances of BEGA are consistent for 100 runs.

Because SGM is an estimation parameter, the computation-time test is necessary to demonstrate the efficiency of BEGA. In Table 2, BEGA was first in 8 benchmarks, and second in other 4 benchmarks. This implies that intensification method is effective to improve the convergence speed.

The objective of BEGA is only to improve the best individual fitness, and BEGA does not consider other fitness-related indexes. This is the difference of BEGA. As shown in Fig. 11, the fitness-mean and fitness-minimum curves of BEGA were not better than others. The reason is that diversity subpopulation always tries to increase Hamming distances between themselves and the current population. This indicates that the balanced-evolution strategy is effective.

4.3 Experiments on different parameters

4.3.1 Different scale problems

In Fig. 12, we demonstrated the robustness of BEGA for different scale problems. With coding length increasing, the best individual fitness of BEGA became much better than others. This indicates that the advantage of SGM as an

Table 1 Used parameter settings in this paper

Items	Parameters
CHC	Population size $P = 90$, the convergence threshold of initialization = 50
DPGA	The main population size = 30, the reserve population size = 60, $p_c = 1$, $p_m = 1/N$, $\alpha = 0.2$, $\tau_1 = 50$, $\tau_2 = 10$
VDMGA	Population size $P = 90$, the convergence threshold of initialization = 0.25 N, $p_m = 0.005$
QEA	Population size $P = 90$, the increment of the rotation angle *Delta* decreased linearly from 0.1 in the first generation to 0.005 in the last generation
GGA	Population size $P = 90$, $p_c = 1$, $p_m = 0.03$
BEGA	Population size $P = 90$, the size of elitism population $P_e = 15$, shift limit of population diversity $D_{sl} = 0.075$, the minimal distance of two adjacent individuals in elitism population $m_d = 3$, the multiplier factor of mutation operator $m_s = 5$
Initial population	In each run, initial population that was randomly generated was the same for all algorithms
Termination condition	In effectiveness test of Sect. 4.2 and the test of Sect. 4.3, termination condition was 600 generations. In efficiency test of Sect. 4.2, fitness threshold was the main termination condition, and maximal generation was the default termination condition
Program and computer	We implemented all algorithms in C++, used complementary multiply with carry generators, and executed on Intel (R) Core (TM) i7 CPU M 620, 4G DDR, Visual Studio 2005, and Windows XP

Table 2 Effectiveness results: MEAN and STD are the mean and standard deviation of the best individual fitness for 100 runs, respectively. Efficiency results: ACT is the average computation time for 20 runs

Problem	Algorithm	Effectiveness results		Efficiency results	
		MEAN	STD	Termination condition	ACT(s)
Trap problem $N = 2160$ Optimal value: 13,200	BEGA	11,669.96	70.55	Fitness threshold = 0.8 × optimal value = 10,560 Maximal generation = 6000	5.057
	CHC	7115.66	94.64		53.336
	DPGA	11,404.56	62.39		6.339
	VDMGA	11,406.36	52.50		4.090
	QEA	11,556.78	70.40		26.873
Deceptive problem $N = 2700$, order-3 Optimal value: 27,000	BEGA	25,068.68	64.03	Fitness threshold = 0.9 × optimal value = 24,300 Maximal generation = 6000	2.280
	CHC	17,108.20	171.59		178.841
	DPGA	24,414.64	80.00		3.070
	VDMGA	22,734.64	132.47		3.954
	QEA	24,286.74	103.83		5.693
Deceptive problem $N = 3000$, order-4 Optimal value: 22,500	BEGA	20,445.28	53.43	Fitness threshold = 0.9 × optimal value = 20,250 Maximal generation = 6000	28.569
	CHC	13,766.70	148.62		192.752
	DPGA	19,500.94	81.29		30.800
	VDMGA	18,994.56	80.08		25.182
	QEA	19,565.20	74.15		158.517
Partially deceptive problem $N = 3000$, order-4 Optimal value: 22,500	BEGA	20,361.60	102.77	Fitness threshold = 0.9 × Optimal value = 20,250 Maximal generation = 6000	31.955
	CHC	13,737.98	150.30		197.607
	DPGA	19,184.74	113.63		40.047
	VDMGA	19,572.96	92.29		21.525
	QEA	19,606.74	98.13		140.658
Overlapping deceptive problem $N = 2002$, order-3 2-bit overlapping Optimal value: 2000	BEGA	1693.17	8.89	Fitness threshold = 0.9 × optimal value = 1800 Maximal generation = 30,000	130.130
	CHC	1283.73	11.94		393.398
	DPGA	1650.34	8.16		136.123
	VDMGA	1683.03	10.61		290.083
	QEA	1659.46	8.59		392.193
PPeaks problem $N = 900$, 250 peaks Optimal value: 1	BEGA	0.99,998	0.00,011	Fitness threshold = 0.98 × optimal value = 0.98 Maximal generation = 6000	47.874
	CHC	0.61,657	0.01,205		1315.310
	DPGA	0.99860	0.00102		63.902
	VDMGA	0.99777	0.00080		33.030
	QEA	0.99,905	0.00,109		334.814

Table 2 (continued)

Problem	Algorithm	Effectiveness results		Efficiency results	
		MEAN	STD	Termination condition	ACT(s)
Binary knapsack problem $N = 2000$ uncorrelated datasets Optimal value: not known	BEGA	82,102.52	48.94	Fitness threshold = 75,000 Maximal generation = 6000	8.791
	CHC	64,247.77	479.75		57.816
	DPGA	81,659.39	84.35		17.884
	VDMGA	81,555.40	58.53		10.304
	QEA	81,738.21	86.51		22.812
Binary knapsack problem $N = 2000$ weakly correlated datasets Optimal value: not known	BEGA	54,404.75	15.34	Fitness threshold = 53,800 Maximal generation = 6000	12.581
	CHC	51,919.43	90.99		126.943
	DPGA	54,304.67	28.67		39.196
	VDMGA	54,357.27	13.19		18.408
	QEA	54,394.55	16.84		34.721
Binary knapsack problem $N = 2000$ strongly correlated datasets Optimal value: not known	BEGA	63,509.84	34.45	Fitness threshold = 63,000 Maximal generation = 6000	9.267
	CHC	60,647.66	70.75		60.086
	DPGA	63,126.22	50.78		30.327
	VDMGA	63,466.24	18.12		10.427
	QEA	63,447.13	26.21		23.646
Bounded knapsack problem $N = 2000$, $b_{\max} = 4$ uncorrelated datasets Optimal value: not known	BEGA	191,858.65	637.57	Fitness threshold = 190,000 Maximal generation = 6000	9.528
	GGA	175,816.93	856.70		48.777
Bounded knapsack problem $N = 2000$, $b_{\max} = 4$ weakly correlated datasets Optimal value: not known	BEGA	136,387.99	74.38	Fitness threshold = 134,000 Maximal generation = 6000	4.576
	GGA	133,640.73	126.56		43.540
Bounded knapsack problem $N = 2000$, $b_{\max} = 4$ strongly correlated datasets Optimal value: not known	BEGA	158,442.11	106.49	Fitness threshold = 157,000 Maximal generation = 6000	10.606
	GGA	155,084.59	107.93		46.171

We use different fitness thresholds as follows. In known-optimal-value problems, there are different threshold coefficients (i.e. 0.8, 0.9, and 0.98). The reason is the different shapes of trap, deceptive-based, and PPeaks problems, as shown in Appendix 1. In unknown-optimal-value problems (i.e. binary and bounded knapsack problems), the fitness thresholds are all less than MEANs of DPGA, VDMGA, QEA, and BEGA, because we do not consider MEANs of CHC and CGA. In addition, maximal generation is 30,000 for overlapping deceptive problem. The reason is that CHC will quickly finish than others, when maximal generation is 6000 (i.e. default termination condition is satisfied first)

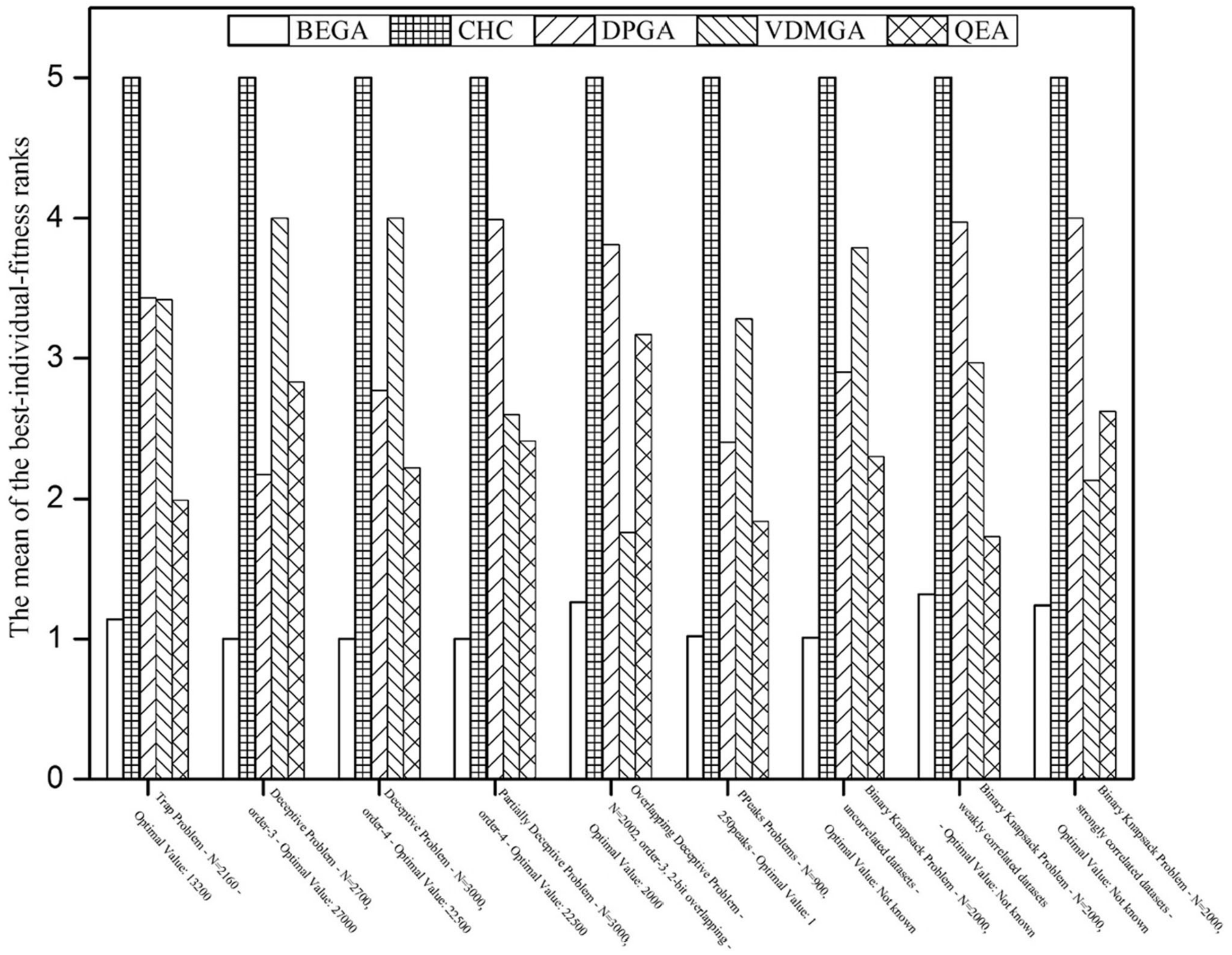


Fig. 10 The mean of the best-individual-fitness ranks for 100 runs in binary-coding problems

estimation parameter becomes obvious, when coding length increases.

4.3.2 Different population sizes

In Fig. 13, we demonstrated the robustness of BEGA for different population sizes. Different population sizes had a little effect on the robustness of BEGA. For instance, there were 3.9% differences of BEGA between population size $P = 90$ and population size $P = 40$ in Fig. 13a.

4.3.3 Different shift limit of population diversities

In Fig. 14, we demonstrated the search-region-control function of D_{sl} for diversity subpopulation in the second stage. When D_{sl} was respectively set to 0.075 and 0.35, the fitness and LDI curves simultaneously shifted in *50*th generation. Fitness-curve and LDI-curve gaps indicate that the ranges of two *od*-Regions for $D_{sl} = 0.075$ and $D_{sl} = 0.35$ are entirely different. This implies that the search-region-control function of D_{sl} is effective for diversity subpopulation.

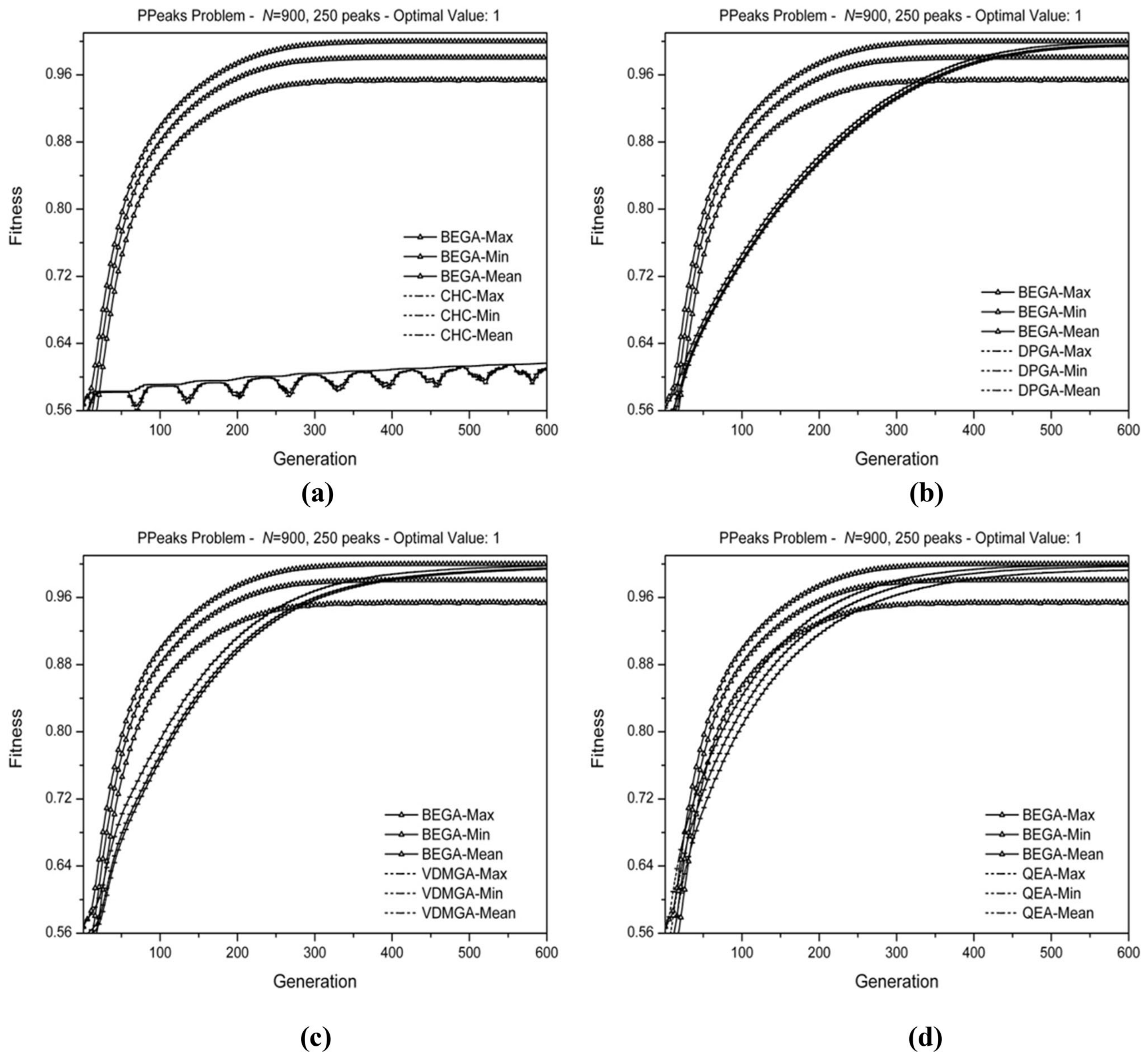


Fig. 11 The fitness curves of PPeaks problem for 100 runs to demonstrate that the objective of BEGA is only to improve the best individual fitness

5 Conclusions

In BEGA, the balanced-evolution strategy realizes the population-evolution balance between the exploration of diversity subpopulation and the exploitation of intensification subpopulation. From a coding-space perspective, this case study solves first and second sub-problems in Sect. 1, which also demonstrates the feasibility of the feedback control scheme. It is worth noting that BEGA is able to directly solve bounded knapsack problem (i.e. symbol-coding problem) as one EA-based solver, and does not transform bounded knapsack problem into an equivalent binary knapsack problem.

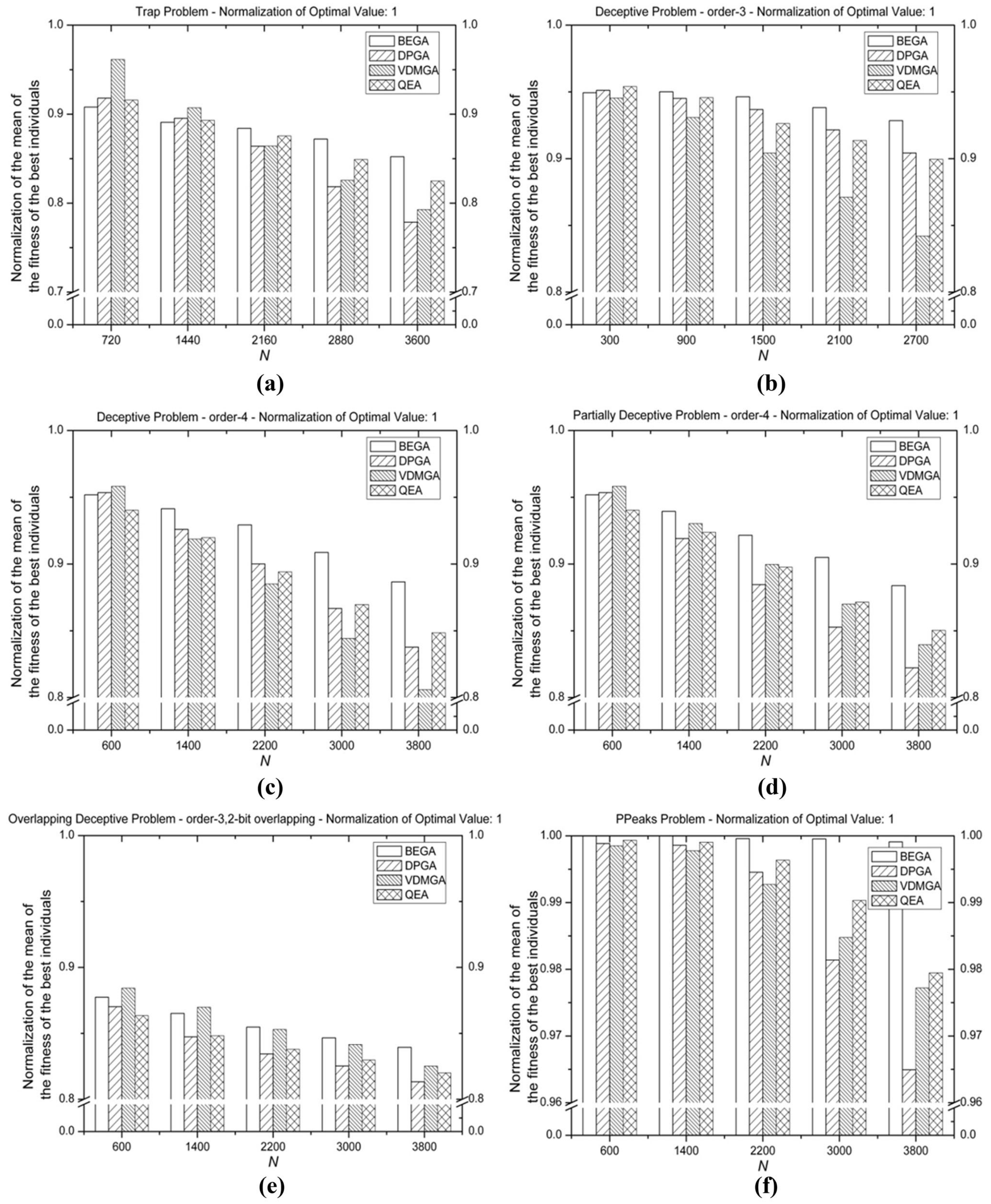


Fig. 12 100-run results to demonstrate the robustness of BEGA for different scale problems (*N* is the coding length)

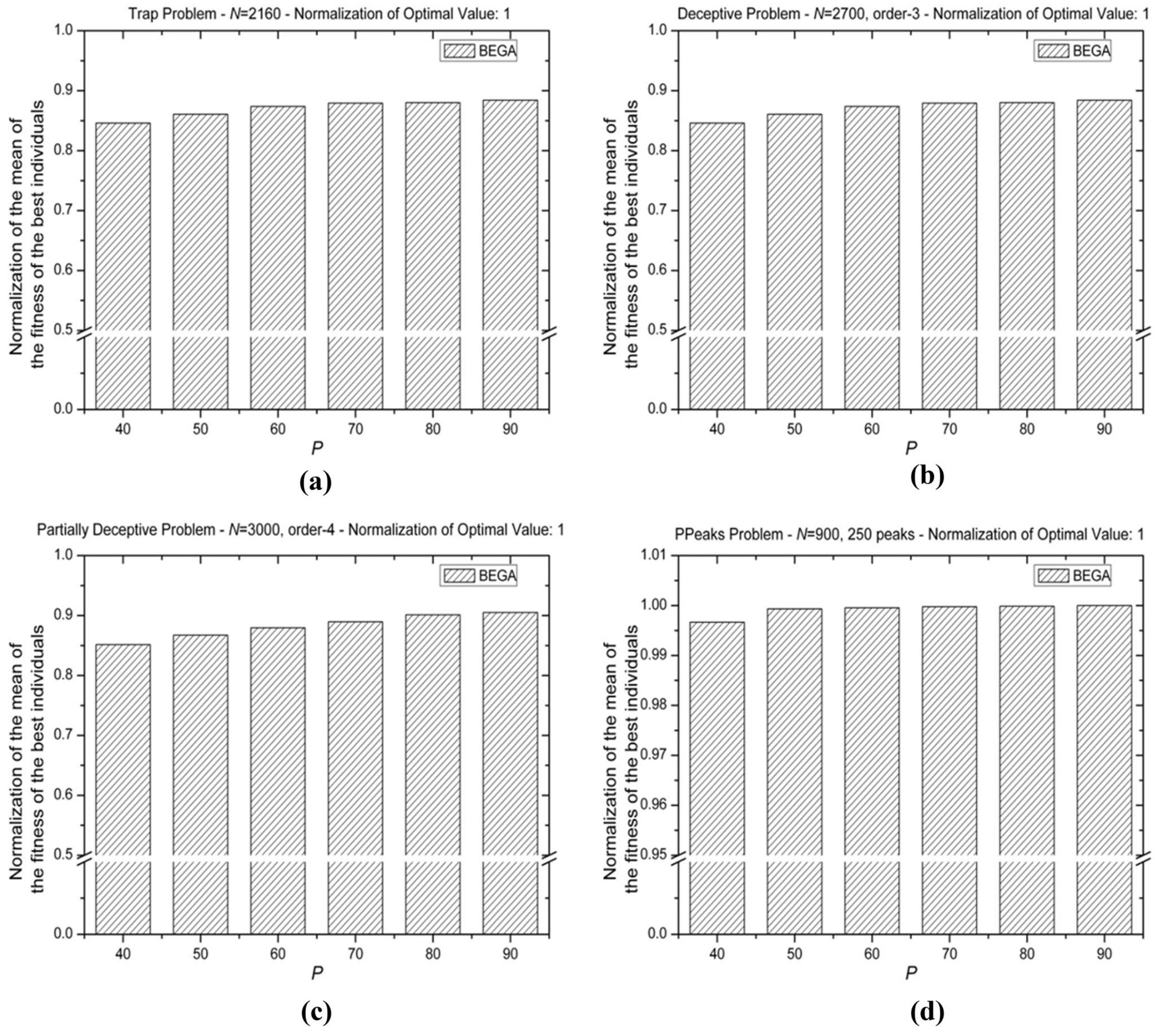


Fig. 13 100-run results to demonstrate the robustness of BEGA for different population sizes (*P* is population size)

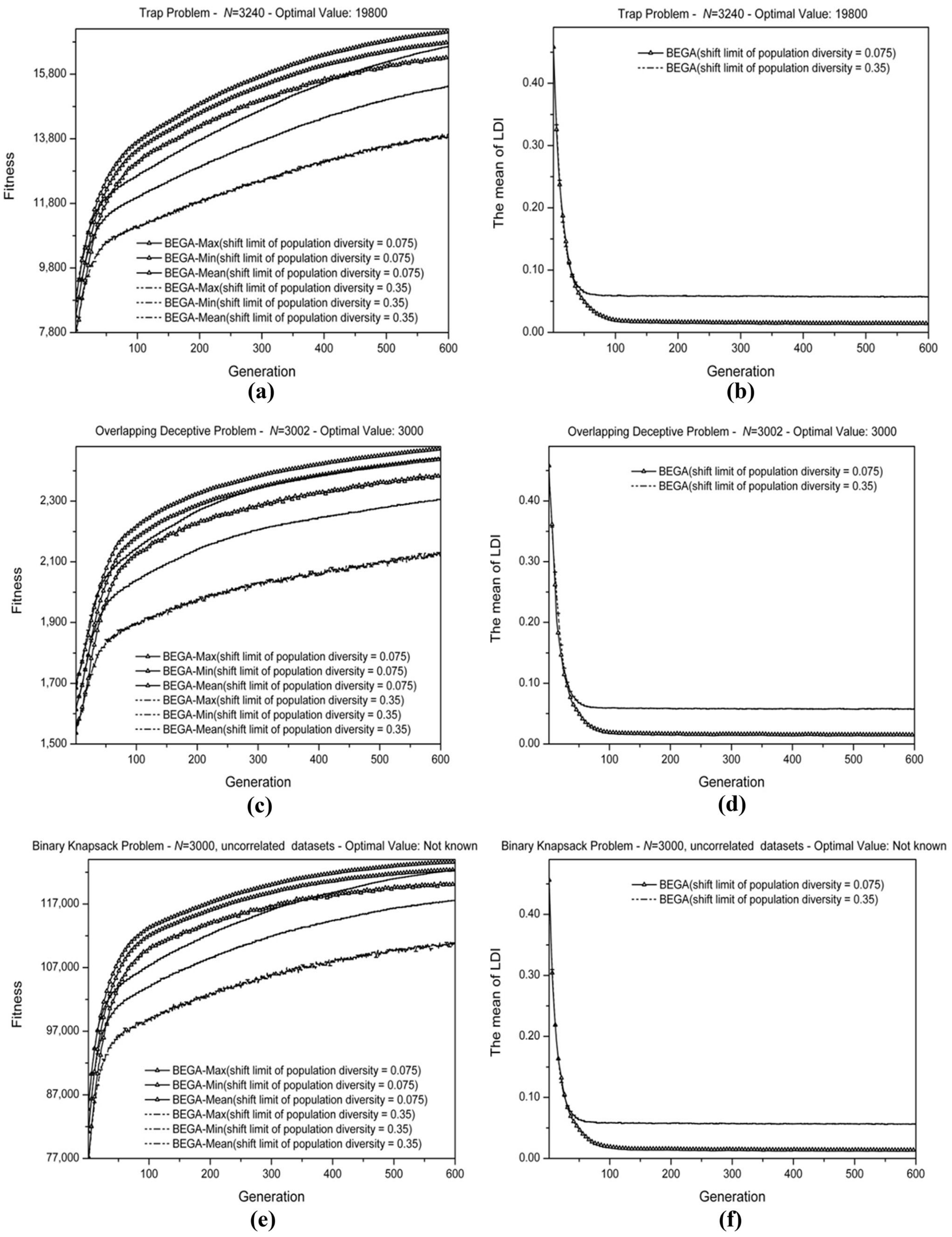


Fig. 14 5-run results to demonstrate the search-region-control function of D_{sl} for diversity subpopulation

Acknowledgements The authors would like to thank anonymous reviewers for their constructive comments, especially for improving the concepts of similarity guide matrix and linear diversity index. This work was supported by National Natural Science Foundation of China (Grant No. 61272518) and YangFan Innovative and Entrepreneurial Research Team Project of Guangdong Province.

Appendix 1

Trap problem includes k basic functions, whose fitness is equal to the fitness sum of k basic functions (García-Martínez and Lozano 2008). The best solution of a basic function, with all ones, has a fitness value of 220 (Table 3). The basic function is defined by

$$f(X) = \sum_{i=0}^{3} F_3(X_{[3i:3i+2]}) + \sum_{i=0}^{5} F_2(X_{[12+2i:13+2i]}) + \sum_{i=0}^{11} F_1(X_{[24+i]}) \tag{21}$$

Analogous to trap problem, basic functions in order-3, order-4, and partially deceptive problems (García-Martínez and Lozano 2008; Baluja 1992) are given in Tables 4, 5, and 6, respectively.

Overlapping deceptive problem (Pelikan et al. 2000) is defined by

$$f(X) = \sum_{i=1}^{N-2} f_d(X_{[i:i+2]}) \tag{22}$$

$$f_d(X_{[i:i+2]}) = \begin{cases} 0.9 & u = 0 \\ 0.8 & u = 1 \\ 0 & u = 2 \\ 1 & u = 3 \end{cases} \tag{23}$$

where i is the first position of each substring $X_{[i:i+2]}$, N is coding length, and u is the number of ones in the substring $X_{[i:i+2]}$.

PPeaks problem (Spears 2000), whose optimal value is 1.0, is defined by

$$f(X) = \frac{1}{N} \max_{i=0}^{p-1} \{N - Hamdis(X, Peak_i)\} \tag{24}$$

where $Hamdis()$ returns Hamming distance between X and $Peak_i$ (i.e. a N-bit string).

Binary knapsack problem is as follow. Let p_j be the profit of type-j item, let w_j be the weight of type-j item, and C is the weight capacity of the knapsack. $X = \{x_1, x_2, \ldots x_j, \ldots x_N\}$ is a binary decision variable. If type-j item is loaded in the knapsack, $x_j = 1$. Otherwise, $x_j = 0$. Binary knapsack problem is defined by using Eqs. (25) and (27). First, we use the methods of generating uncorrelated, weakly correlated, and strongly correlated datasets (Martello et al. 1999; Pisinger 1999; Truong et al. 2013). Uncorrelated dataset: p_j and w_j are randomly distributed in (10, R). Weakly correlated dataset: w_j is randomly distributed in (1, R), and p_j ($p_j \geq 1$) is randomly distributed in ($w_j - R/10$, $w_j + R/10$). Strongly correlated dataset: w_j is randomly distributed in (1, R), and p_j is $w_j + 10$. In this paper, R is 100. Secondly, we use the constraint handling method (Zitzler 1999) as follows. Items with the lowest profit/weight ratio q_j (i.e. $q_j = p_j/w_j$ $1 \leq j \leq N$) are removed first. Items are removed one by one, until the capacity constraint is satisfied.

Table 3 Basic functions in trap problem. ONESUM is the number of bits, whose value is equal to one

ONESUM	0	1	2	3
F_3	4	2	0	10
F_2	5	0	10	
F_1	0	10		

Table 4 Order-3 deceptive problem

X	000	001	010	100	110	011	101	111
$f(X)$	28	26	22	14	0	0	0	30

Table 5 Order-4 deceptive problem

X	0000	0001	0010	0100	1000	0011	0101	0110	1001	1010	1100	1110	1101	1011	0111	1111
$f(X)$	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0	30

Table 6 Order-4 partially deceptive problem

X	0000	0001	0010	0100	1000	0011	0101	0110	1001	1010	1100	1110	1101	1011	0111	1111
$f(X)$	28	26	24	22	20	18	30	14	12	10	8	6	4	2	0	16

$$\mathit{Maximize} \sum_{j=1}^{N} p_j x_j \tag{25}$$

$$C = 0.5 \sum_{j=1}^{N} w_j \tag{26}$$

subject to

$$\sum_{j=1}^{N} w_j x_j \le C \tag{27}$$

Bounded knapsack problem is also formulated by using Eqs. (25) and (27). The difference of bounded knapsack problem is that x_j expresses how many type-j item is loaded in the knapsack. First, we also use the same methods of generating test datasets (Martello et al. 1999; Pisinger 1999) for bounded knapsack problem. Secondly, similar to binary knapsack problem, the difference of bounded knapsack problem is that the constraint handling method gradually decreases the number of each item. In this paper, $b_{\max}$ is 4 for bounded knapsack problem. According to $1 \le b_j \le b_{\max}$, b_j is randomly generated. We assume that p_j, w_j, b_j, and C are greater than 0 and

$$\sum_{j=1}^{N} w_j b_j > C \tag{28}$$

$$w_j b_j \le C \quad 1 \le j \le N \tag{29}$$

$$C = 0.5 \sum_{j=1}^{N} w_j b_j \tag{30}$$

Appendix 2

CHC uses cross generational elitist selection, heterogeneous recombination, and cataclysmic mutation (Eshelman 1991). Two parents are only allowed to mate, when Hamming distance between two parents is greater than the threshold. CHC only carries out mutation to reinitialize the population by keeping the best individual, when the threshold drops to zero.

DPGA (Park and Ryu 2010) and VDMGA (Fernandes and Rosa 2006, 2008) are given in Algorithms 5 and 6, respectively.

Algorithm 5 Dual-population genetic algorithm

```
1:  Input: the size of main population, the size of reserve population, crossover probability $p_c$, mutation probability $p_m$,
2:         coding length $N$, the parameters of adjusting $\delta(\alpha, \tau_1, and\ \tau_2)$
3:  Step 1: Initialize main population $M_P$ and reserve population $R_P$. Evaluate main population by using the fitness
4:          function, and evaluate reserve population by using the distance fitness function.
5:  Step 2: Based on main population, breed the inbred offspring $I_M$ by using two-point crossover and bitwise
6:          mutation. Based on reserve population, breed the inbred offspring $I_R$.
7:  Step 3: Based on main population and reserve population, breed the crossbred offspring $C$.
8:  Step 4: $O_M = I_M \cup C$ . $O_R = I_R \cup C$. According to the fitness function, evaluate $O_M$. According to the distance fitness
9:          function, evaluate $O_R$.
10: Step 5: Update main population by using survival selection. Then, replace reserve population with $O_R$.
11: Step 6: Check whether termination condition is satisfied or not. If it is not satisfied, go to Step 2.
12: Step 7: The individual with the best fitness represents the final solution.
```

Algorithm 6 Variable dissortative mating genetic algorithm

```
1:  Input: population size $P$, coding length $N$, the convergence threshold of initialization, mutation probability $p_m$
2:  Step 1: Initialize a population, and evaluate this population.
3:  Step 2: Breed the new offspring as follows.
4:          do{   for( i = 1 to $P/2$)
5:                {   Randomly select two parents. Then, compute their Hamming distance $H_D$.
6:                    If($H_D \geqslant threshold$)   carry out uniform crossover and bit-flip mutation.   }
7:                if( the number of failed matings > the number of successful matings) threshold = threshold - 1;
8:                else                                                          threshold = threshold + 1;
9:        }while( successful matings == 0 )
10: Step 3: Evaluate the new offspring.
11: Step 4: Carry out elite selection for the population and the new offspring, and choose the new population.
12: Step 5: Check whether termination condition is satisfied or not. If it is not satisfied, go to Step 2.
13: Step 6: The individual with the best fitness represents the final solution.
```

QEA (Han and Kim 2002, Platel et al. 2009) is inspired by the principle of quantum computing. Quantum gate $U(\theta)$ is given in Eq. (31), and θ is equal to $s(\alpha_j\beta_j) \times Delta$ in Table 7.

$$U(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{31}$$

In the bounded-knapsack-problem (i.e. symbol-coding problem) field, dynamic programming, branch-and-bound algorithm, and reduction algorithm are frequently used (Martello and Toth 1990). Another kind of algorithm is to transform bounded knapsack problem into an equivalent binary knapsack problem (Martello and Toth 1990). However, this implies much more computation cost, because coding length increases. In EAs, there is little research, which directly solves bounded knapsack problem. Thus, GGA is used as the compared algorithm of bounded knapsack problem.2

Table 7 Lookup table of θ

x_j	e_j	$f(X) \geq f(E)$	Delta	$s(\alpha_j\beta_j)$			
				$\alpha_j\beta_j > 0$	$\alpha_j\beta_j < 0$	$\alpha_j = 0$	$\beta_j = 0$
0	0	–	0	–	–	–	–
1	1	–	0	–	–	–	–
0	1	False	Delta	+ 1	− 1	0	± 1
0	1	True	Delta	− 1	+ 1	± 1	0
1	0	False	Delta	− 1	+ 1	± 1	0
1	0	True	Delta	+ 1	− 1	0	± 1

$f()$ is the fitness function. $s(\alpha_j\beta_j)$ is the sign of θ. e_j and x_j are the jth bits of the best solution E and a binary decision variable X, respectively

Algorithm 7 Greedy genetic algorithm

```
1:  Input: population size $P$, coding length $N$, crossover probability $p_c$, mutation probability $p_m$
2:  Step 1: Initialize a population. Evaluate this population, and simultaneously perform the greedy constraint handling
3:          as follows. The processed-item order is determined by the profit/weight ratio and the value of each $x_j$.
4:          First, the capacity is set to 0, and process each $x_j$ in the descending order. Secondly, if and only if $x_j$ is not
5:          equal to 0, $x_j$ increases from 1 to $b_j$, until the capacity constraint is violated.
6:  Step 2: Perform the greedy roulette wheel operator as follows. Copy 40% of the best individuals into the next
7:          generation. Then, randomly choose 1/3 of the remaining 60% individuals to the next generation. Finally,
8:          by using roulette wheel selection, choose 40% individuals, which participate in one point crossover.
9:  Step 4: Breed the new offspring by using one point crossover. Perform the greedy mutation operator as follows. If
10:         the chosen genome $x_j$ is equal to 0, $x_j$ is set to $b_j$. Otherwise, $x_j$ is set to 0.
11: Step 5: Evaluate the new population, and simultaneously perform the greedy constraint handling.
12: Step 6: Check whether termination condition is satisfied or not. If it is not satisfied, go to Step 3.
13: Step 7: The individual with the best fitness represents the final solution.
```

References

Alba E, Dorronsoro B (2005) The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans Evol Comput 9:126–142. https://doi.org/10.1109/TEVC.2005.843751

Baluja S (1992) A massively distributed parallel genetic algorithm. Tech. Rep. No. CMU-CS- 92-196R. Carnegie Mellon University

Burke E, Gustafson S, Kendall G, Krasnogor N (2002) Advanced population diversity measures in genetic programming. In: Proceedings of parallel problem solving from nature. Springer, pp 341–350. https://doi.org/10.1007/3-540-45712-7_33

Burke E, Gustafson S, Kendall G (2004) Diversity in genetic programming: an analysis of measures and correlation with fitness. IEEE Trans Evol Comput 8:47–62. https://doi.org/10.1109/TEVC.2003.819263

Cao ZJ, Shi YH, Rong XF, Liu BL, Du ZQ, Yang B (2015) Random grouping brain storm optimization algorithm with a new dynamically changing step size. In: Proceedings of the International Conference on Swarm Intelligence, Lecture Notes in Computer Science. Springer, pp 357–364. https://doi.org/10.1007/978-3-319-20466-6_38

Chen G, Low CP, Yang ZH (2009) Preserving and exploiting genetic diversity in evolutionary programming algorithms. IEEE Trans Evol Comput 13:661–673. https://doi.org/10.1109/TEVC.2008.2011742

Črepinšek M, Liu SH, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput Surv 45:1–33. https://doi.org/10.1145/2480741.2480752

Darwen PJ, Yao X (2001) Why more choices cause less cooperation in iterated prisoner's dilemma. In: Proceedings of the 2001 Congress on Evolutionary Computation. IEEE, pp 987–994. https://doi.org/10.1109/cec.2001.934298

De Jong K (1975) An analysis of the behavior of a class of genetic adaptive systems. Dissertation, University of Michigan

De Jong K (2007) Parameter setting in EAs: a 30 year perspective. In: Lobo FG, Lima CF, Michalewicz Z (eds) Parameter setting in evolutionary algorithms. Springer, Berlin, pp 1–18

Dekkers A, Aarts E (1991) Global optimization and simulated annealing. Math Program 50:367–393. https://doi.org/10.1007/BF01594945

Eiben AE, Schippers C (1998) On evolutionary exploration and exploitation. Fundam Inform 35:35–50. https://doi.org/10.3233/FI-1998-35123403

Eiben AE, Smit SK (2011) Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol Comput 1:19–31. https://doi.org/10.1016/j.swevo.2011.02.001

Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms. IEEE Trans Evol Comput 3:124–141. https://doi.org/10.1109/4235.771166

Eshelman LJ (1991) The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In: Gregory JR (ed) Foundations of genetic algorithms, vol 1. Morgan Kaufmann Publishers Inc, San Francisco, pp 265–283

Fernandes C, Rosa A (2006) Self-regulated population size in evolutionary algorithms. In: Runarsson TP, Beyer HG et al (eds) Parallel problem solving from nature–PPSN IX. Springer, Iceland, pp 920–929

Fernandes C, Rosa A (2008) Self-adjusting the intensity of assortative mating in genetic algorithms. Soft Comput 12:955–979. https://doi.org/10.1007/s00500-007-0265-9

García-Martínez C, Lozano M (2008) Local search based on genetic algorithms. In: Siarry P, Michalewicz Z (eds) Advances in metaheuristics for hard optimization. Springer, Berlin, pp 199–221

Glover F (1997) Heuristics for integer programming using surrogate constraints. Decis Sci 8:156–166. https://doi.org/10.1111/j.1540-5915.1977.tb01074.x

Han KH, Kim JH (2002) Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans Evol Comput 6:580–593. https://doi.org/10.1109/TEVC.2002.804320

Harik GR, Lobo FG, Goldberg DE (1999) The compact genetic algorithm. IEEE Trans Evol Comput 3:287–297. https://doi.org/10.1109/4235.797971

Holland JH (1975) Adaptation in natural and artificial systems. The MIT Press, Ann Arbor

Jaccard P (1912) The distribution of the flora in the alpine zone. New Phytol 11:37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x

Koumousis VK, Katsaras CP (2006) A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans Evol Comput 10:19–28. https://doi.org/10.1109/TEVC.2005.860765

Lacevic B, Amaldi E (2010) On population diversity measures in euclidean space. In: Proceedings of the 2010 Congress on Evolutionary Computation. IEEE, pp 1–8. https://doi.org/10.1109/cec.2010.5586498

Lacevic B, Konjicija S, Avdagic Z (2007) Population diversity measure based on singular values of the distance matrix. In: Proceedings of the 2007 Congress on Evolutionary Computation. IEEE, pp 1863–1869. https://doi.org/10.1109/cec.2007.4424700

Lee CY, Yao X (2004) Evolutionary programming using mutations based on the levy probability distribution. IEEE Trans Evol Comput 8:1–13. https://doi.org/10.1109/TEVC.2003.816583

Lozano M, Herrera F, Cano JR (2008) Replacement strategies to preserve useful diversity in steady-state genetic algorithms. Inf Sci 178:4421–4433. https://doi.org/10.1016/j.ins.2008.07.031

Martello S, Toth P (1990) Knapsack problems: algorithms and computer implementations. Wiley, Hoboken

Martello S, Pisinger D, Toth P (1999) Dynamic programming and strong bounds for the 0–1 knapsack problem. Manag Sci 45:414–424. https://doi.org/10.1287/mnsc.45.3.414

Mattiussi C, Waibel M, Floreano D (2004) Measures of diversity for populations and distances between individuals with highly reorganizable genomes. Evol Comput 12:495–515. https://doi.org/10.1162/1063656043138923

McGinley B, Maher J, O'Riordan C, Morgan F (2011) Maintaining healthy population diversity using adaptive crossover, mutation, and selection. IEEE Trans Evol Comput 15:692–714. https://doi.org/10.1109/TEVC.2010.2046173

Morrison RW, De Jong K (2002) Measurement of population diversity. In: Collet P, Fonlupt C, Hao J, Lutton E, Schoenauer M (eds) Artificial evolution. Springer, France, pp 31–41

Park T, Ryu KR (2010) A dual-population genetic algorithm for adaptive diversity control. IEEE Trans Evol Comput 14:865–884. https://doi.org/10.1109/TEVC.2010.2043362

Pelikan M, Goldberg DE, Cantú-paz EE (2000) Linkage problem, distribution estimation, and bayesian networks. Evol Comput 8:311–340. https://doi.org/10.1162/106365600750078808

Pisinger D (1999) Core problems in knapsack algorithms. Oper Res 47:570–575. https://doi.org/10.1287/opre.47.4.570

Platel MD, Platel MD, Schliebs S, Schliebs S, Kasabov N, Kasabov N (2009) Quantum-inspired evolutionary algorithm: a multimodel EDA. IEEE Trans Evol Comput 13:1218–1232. https://doi.org/10.1109/TEVC.2008.2003010

Resende MGC, Ribeiro CC, Glover F, Martí R (2010) Scatter search and path-relinking: fundamentals, advances, and applications. In: Gendreau M, Potvin JY (eds) Handbook of metaheuristics. Springer, Berlin, pp 87–107

Shi YH (2015) Brain storm optimization algorithm in objective space. In: IEEE Congress on Evolutionary Computation. IEEE, pp 1227–1234. https://doi.org/10.1109/cec.2015.7257029

Smit SK, Eiben AE (2009) Comparing parameter tuning methods for evolutionary algorithms. In: Proceedings of the 2009 Congress on Evolutionary Computation. IEEE, pp 399–406. https://doi.org/10.1109/cec.2009.4982974

Spears WM (2000) Evolutionary algorithms: the role of mutation and recombination. Springer, Berlin

Truong TK, Li KL, Xu YM (2013) Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem. Appl Soft Comput 13:1774–1780. https://doi.org/10.1016/j.asoc.2012.11.048

Ursem RK (2002) Diversity-guided evolutionary algorithms. In: Guervós JJM, Adamidis P et al (eds) Parallel problem solving from nature—PPSN VII. Springer, Espana, pp 462–471

Zhan ZH, Zhang J, Shi YH, Liu HL (2012) A modified brain storm optimization. In: IEEE Congress on Evolutionary Computation. IEEE, pp 1–8. https://doi.org/10.1109/cec.2012.6256594

Zhang J, Chung HSH, Lo W (2007) Clustering-based adaptive crossover and mutation probabilities for genetic algorithms. IEEE Trans Evol Comput 11:326–335. https://doi.org/10.1109/TEVC.2006.880727

Zhu HY, Shi YH (2015) Brain storm optimization algorithms with K-medians clustering algorithms. In: International Conference on Advanced Computation Intelligence. IEEE, pp 107–110. https://doi.org/10.1109/icaci.2015.7184758

Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: methods and applications. Dissertation, Swiss Federal Institute of Technology Zurich