Applied Soft Computing 12 (2012) 1693–1707

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Diversity Guided Evolutionary Programming: A novel approach for continuous optimization

Mohammad Shafiul Alam[a,*], Md. Monirul Islam[a], Xin Yao[b,c], Kazuyuki Murase[d]

[a] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh
[b] Nature Inspired Computation and Applications Laboratory, Department of Computer Science, University of Science and Technology of China, Hefei, Anhui 230027, China
[c] Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, The University of Birmingham, UK
[d] Department of Human and Artificial Intelligence Systems, University of Fukui, Fukui 910-8507, Japan

ARTICLE INFO

Article history:
Received 9 March 2011
Received in revised form 16 August 2011
Accepted 2 February 2012
Available online 28 February 2012

Keywords:
Evolutionary programming
Population diversity
Exploitation and exploration
Continuous optimization

ABSTRACT

Avoiding premature convergence to local optima and rapid convergence towards global optima has been the major concern with evolutionary systems research. In order to avoid premature convergence, sufficient amount of genetic diversity within the evolving population is considered necessary. Several studies have focused to devise techniques to control and preserve population diversity throughout the evolution. Since mutation is the major operator in many evolutionary systems, such as evolutionary programming and evolutionary strategies, a significant amount of research has also been done for the elegant control and adaptation of the mutation step size that is proper for traversing across the locally optimum points and reach for the global optima. This paper introduces Diversity Guided Evolutionary Programming, a novel approach to combine the best of both these research directions. This scheme incorporates diversity guided mutation, an innovative mutation scheme that guides the mutation step size using the population diversity information. It also takes some extra diversity preservative measures to maintain adequate amount of population diversity in order to assist the proposed mutation scheme. An extensive simulation has been done on a wide range of benchmark numeric optimization problems and the results have been compared with a number of recent evolutionary systems. Experimental results show that the performance of the proposed system is often better than most other algorithms in comparison on most of the problems.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Different major branches of evolutionary algorithms (EAs) have evolved during the last few decades, such as genetic algorithms (GAs) [1], evolutionary programming (EP) [2], evolution strategies (ESs) [3] and their many variants. They have been extensively used with wide range of problems, like function optimization (e.g. [4–6]), multi-objective optimization (e.g. [7–9]), discrete optimization (e.g. [10,11]), learning decision rules (e.g. [12]), optimizing object recognition model (e.g. [13]), evolving fuzzy rules (e.g. [9,14]) and training neural networks (e.g. [15,16]). EAs are stochastic search methods that work with a population of approximate solutions (i.e., chromosomes) that undergoes an evolutionary process by the application of some operators borrowed from natural genetics, like crossover, mutation, and selection. Crossover and mutation are variation operators applied on existing chromosomes to produce new ones, while selection is used to probabilistically pick better chromosomes for the next generation. This process is repeated again and again until a chromosome with sufficient quality is found or a predefined amount of computational resource is exhausted.

EAs are more resilient against local optima than other greedy and local search algorithms because the population provides the advantage of preserving diversity. Still it is often found that the evolving population loses its genetic diversity too soon and the chromosomes prematurely get trapped around the locally optimal points of the fitness landscape. This usually happens with complex multi-modal problems that have numerous local optima in the fitness landscape. Several research works have been conducted with the problem of premature convergence and loss of diversity. The existing methods can be divided into five categories: (1) dynamic operator and parameter control , (2) complex population structures, (3) specialized selection operators, (4) specialized variation operators, and (5) memory-based algorithms. The evolution of the population is carried out by the genetic operators, i.e., selection, crossover and mutation which are affected by several parameters, such as population size, crossover rate, mutation rate and step size. Dynamic operator and parameter

* Corresponding author.
E-mail address: shuvo23@gmail.com (M.S. Alam).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2012.02.002

control methods attempt to adapt these operators and parameters and thus control the convergence characteristics of the evolving population in order to avoid premature convergence. Some evolutionary systems employ multiple sub-populations or complex population structures, e.g., island [17, C6.4], [18], nation [19], religion [20], which reduce gene flow across the population and promote genetic diversity. The semi-isolated nature of the sub-populations helps the algorithms maintain overall population diversity and avoid premature convergence. There exist a number of methods, e.g. crowding [21], fitness sharing [22–24], that alter the traditional fitness based selection operator to foster multiple diverse species within the population and thus preserve population diversity. The variation operators, i.e., crossover and mutation can also be altered to improve their explorative capacity, as demonstrated in [5,6]. More explorations usually provide better immunity against premature convergence, but at the cost of reduced convergence speed. So a balance between explorations and exploitations is sought by altering the crossover and mutation operators in various ways [25–28]. Memory-based algorithms provide the population with additional diversity information at the cost of additional memory usage. For example, in Diploid GA [29,30] each chromosome has an additional strand of 'repressed' genes that store some extra information and operate as a latent source of useful population diversity for the next generations in order to avoid premature convergence.

Since mutation is the sole variation operator in many evolutionary systems, including EP and ES [31,32] schemes, much research, both practical, e.g. [33] and theoretical, e.g. [34] has been done for improving the effectiveness of mutation as a search mechanism. A small mutation step size provides better search stability, but is likely to get trapped into locally optimal points while a large mutation step size is better immune against local optima, but the search may be unstable with unacceptably slow or no convergence to the global optima. So most existing works [6,35–38] try to adapt the mutation rate and step size dynamically in order to achieve adequate convergence rate and avoid premature convergence to local optima. However, none of the existing schemes, to our best knowledge, makes use of population diversity information to guide the mutation step size. The proposed system, Diversity Guided Evolutionary Programming introduces and incorporates diversity guided mutation, a novel mutation scheme that makes use of the population diversity information in order to adapt the mutation step size suitably to avoid premature convergence. The balance between the exploitative and explorative features of the mutation operation is sought by employing diversity information existent at two different granularities: micro and macro. At the micro level, diversity among genetically very similar chromosomes is used to induce small mutation steps that are suitable for exploitation of the existing chromosomes, while at the macro level the diversity across two random chromosomes is used to introduce relatively large variations that are suitable for search space explorations. To assist the proposed diversity based mutation operator with additional diversity, the selection operator is also altered. In addition to fitness based selection, the selection operator also looks for chromosomes that are either duplicates or have failed to improve for a long time and replace them with randomly placed chromosomes in a way that promotes population diversity. The proposed scheme has been compared with several other existing research works on a wide range of low and high dimensional, unimodal and multimodal, regular, rotated and hybrid composite benchmark functions. Experimental results demonstrate the effectiveness of the proposed scheme as it often outperforms other algorithms on most of the functions.

The rest of this paper is organized as follows. Section 2 reviews related works on maintaining the population diversity and avoiding premature convergence. Section 3 describes the proposed algorithm along with its central component, the diversity guided mutation scheme with the pseudocode. A brief analysis of its strengths and how it differs from other existing works are also presented in this section. Section 4 provides details of the benchmark problems, parameter settings and compares the results with other algorithms. Section 5 presents a short discussion on the results and leaves a few suggestions on future research directions.

2. Related works

Significant amount of research has been done on the issue of maintaining population diversity and avoiding premature convergence. We have reviewed a number of representative research works and categorized them, as already mentioned, in the following five categories: (1) dynamic operator and parameter control, (2) complex population structures, (3) specialized selection operators, (4) specialized variation operators, and (5) memory-based algorithms.

2.1. Dynamic operator and parameter control

The convergence of the population is affected by the synergy of different evolutionary operators, such as crossover and mutation and the values of different parameters, like population size, crossover rate, mutation rate and step size. So these operators and parameter settings can be dynamically adjusted to guide the evolution, as demonstrated by several previous works employing deterministic control [39,40], adaptive control [41], and self-adaptive parameter control [35]. Among them, the simplest scheme is the deterministic control where some predetermined policy controls the operator and parameter settings. Due to the fixed preset policy, it may not cope with the dynamic scenario that evolves continuously during the evolution. So a better alternative is the adaptive control methods that exploit feedback from the population to control the policy. A good example is the Diversity Guided Evolutionary Algorithm (DGEA) [42]. DGEA applies diversity-decreasing operators (selection and recombination) as long as the diversity is above a certain threshold, d_{low}. When the diversity drops below d_{low} the algorithm switches to a diversity-increasing operator (mutation) until another threshold, d_{high} is reached. The remaining scheme, self-adaptive parameter control methods encode the parameter values, e.g., mutation rate and step size, as genes in chromosomes and make them evolve with other regular genes. This ensures a self-adaptive and distributed, rather than global control over the parameter values and the evolutionary process itself adapts and finds its way towards an appropriate settings. However, the effectiveness of self-adaptive control has not yet been demonstrated with a wide range of problems.

2.2. Complex population structures

In simple GA, any chromosome can mate with any other, so an exceptionally superior quality gene may spread rapidly throughout the entire population. Such high gene flow may cause loss of genetic diversity which eventually leads to premature convergence. To address this problem, some algorithms have adopted complex population structures or multiple sub-populations, e.g., the diffusion model [17, C6.3], Island-model GA (IMGA) [17, C6.4], [18] multinational GA [19], religion-based EA [20], forking GA [43], bi-objective multi-population algorithm [4], variable island GA [18] and dual population GA [44]. Periodic migration of chromosomes between the population structures, e.g., islands, nations and religions, facilitates exchange of useful information. The semi-isolated and spatially separated nature of the sub-populations lowers gene flow and promotes overall population diversity. The number of sub-populations is either fixed, as in [17,44], or variable

allowing suitable merging and splitting of the sub-populations, as in [18,19]. Also, the different evolving sub-populations may have the same evolutionary objective, as in [17,19] or completely different objectives, as in [44]. However, the convergence rate and effectiveness of these algorithms is sensitive to several parameters, such as migration rates and size, migration policy, number and topology of the population structures. A number of research works, such as [45–48], have made both theoretical and experimental study on the effect of these parameter.

2.3. Specialized selection operators

Some algorithms, such as crowding [21] and fitness sharing [22–24] alter the selection operator in a way that penalizes similar chromosomes and promotes diverse ones. This eventually forms multiple diverse species within the population, each evolving to a different peak of the fitness landscape. The crowding scheme randomly selects a number, say *CF*, (crowding factor) of chromosomes from the current population after producing an offspring, *O*. The chromosome that is most similar to *O* is then replaced by it. This reduces the possibility of hosting similar chromosomes within the population. Several variants of crowding exist in the literature, such as deterministic crowding [49,50], probabilistic crowding [51], restricted tournament selection (RTS) [52]. Crowding methods are good at locating multiple optima simultaneously.

The other approach, fitness sharing [22–24] is founded upon the idea that there is only limited and fixed amount of 'resources' (i.e., fitness value) available at each neighborhood of the fitness landscape. So, all the chromosomes occupying the same neighborhood have to share the resource/fitness value. Fitness sharing transforms their raw fitness values in such a way that the larger the density of chromosomes in a region, the more penalized their fitness values become. This discourages similar chromosomes within the population and promotes diversity.

Another example of employing specialized selection operator is the Diversity Control Oriented Genetic Algorithm (DCGA) [53]. DCGA introduces cross-generational probabilistic survival selection (CPSS), a novel selection scheme that calculates the survival probability of each chromosome based on its distance from the current best chromosome. To select a chromosome for the next generation, CPSS considers not only its fitness but also its dissimilarity with the current best chromosome in a way that promotes useful genetic diversity.

2.4. Specialized variation operators

The genetic variation operators, i.e., recombination and mutation, have been altered in several ways to alter their capacity of explorations and exploitations. For example, Real Coded Memetic Algorithm (RCMA) with XHC [25] employs crossover hill climbing, an exploitative crossover operator that performs effective local tuning of the chromosomes to improve them. The memetic algorithm component of RCMA ensures high level of population diversity to provide a reliable ground on which the more exploitative crossover operator performs its fine tuning. Some evolutionary systems, such as EP and ES [31,32] schemes employ mutation as their only variation operator. A number of innovative distributions have been proposed for mutation in EP, such as Cauchy distribution [6], a combination of Cauchy and Gaussian distributions [6] and Lévy distribution [5]. These distributions introduce relatively large variations to produce offspring which increases the exploration ability of the mutation operator. Larger variations around the parent chromosomes contribute to more population diversity and better resilience against local optima and premature convergence. Two other algorithms, RCMA with Adaptive Local Search (LSRCMA) [26] and Differential Evolution with Neighborhood Search (NSDE) [27,28] make attempts to combine the benefits of more exploitative local search and neighborhood search techniques with more explorative memetic algorithm and differential evolution respectively. Another approach, Covariance Matrix Adaptation Evolution Strategy (CMAES) [38] continuously adapts the mutation step size in such a way that the likelihood of producing better offspring is maximized. A proper balance between exploitations and explorations is essential to maintain necessary genetic diversity, avoid premature convergence and achieve adequate convergence speed, as demonstrated in [25].

2.5. Memory-based algorithms

Memory-based algorithms make use of an additional explicit or implicit memory in order to maintain more genetic information that provides the population with additional genetic diversity. Examples of memory-based algorithms include GA with unexpressed genes (GAUG) [54], Diploid GA [29,30], Dual GA (DGA) [55,56], Primal–Dual GA (PDGA) [57,58] and so on. The biological inspiration behind memory based algorithms is that most beings in nature have a large number of 'repressed' genes in their chromosomes with relatively small number of dominant genes that are expressed in their characteristics. The repressed genes store large amount of latent information that are used for producing offspring and thus act as a source of population diversity for the next generations. GAUG uses haploid (single strand) chromosomes, but includes a number of unexpressed genes that are used for preserving diversity. Diploid GAs use diploid (double strand) chromosomes and follow some predetermined dominance rules that decide which genes will be expressed and used for fitness evaluation. The remaining genes are used for providing diversity to produce offspring. Both DGA and PDGA employ haploid chromosomes, but the chromosomes may be interpreted in a complemented way to provide additional diversity. In PDGA, a less fit chromosome may be interpreted both as original and as complemented, and the original one is replaced by the complemented one if the latter gives better fitness evaluation. Memory based algorithms are especially suited for dynamic optimization problems because the usage of additional memory for extra diversity makes it easier to adapt to external and unprecedented environmental changes.

3. Diversity Guided Evolutionary Programming (DGEP)

DGEP differs from the Classical Evolutionary Programming (CEP) [2,59] at two different points. First, DGEP introduces and incorporates diversity guided mutation (DGM), a novel mutation scheme that uses population diversity information to adapt the mutation step size. Second, DGEP employs a few simple diversity preserving schemes to assist the DGM scheme with additional amount of diversity as sufficient diversity is required by DGM to perform both exploitative and explorative search around the current chromosomes. DGM makes use of the population diversity information at two different levels: diversity among very similar chromosomes is used for exploitations of existing solutions while diversity among dissimilar chromosomes across the population is used for search space explorations. After random initialization and during early generations, both these diversity values (say, div_{low} and div_{high}) are usually large. As the population converges around locally optimum points of the search space, div_{low} drops rapidly, but div_{high} usually remains sufficiently high because of the existence of several locally optimal points within the population. This high amount of diversity among non-neighbors facilitates large mutations and thus helps avoiding premature convergence around any locally optimum point. Gradually the entire population converges toward a single global optimum which causes both div_{low} and div_{high} to

decrease and allows the algorithm to converge. In addition to the DGM scheme for more effective mutations, DGEP also employs some diversity preserving mechanisms to help the DGM scheme, such as elimination of identical or nearly duplicate chromosomes, detection and replacement of chromosomes that have reached fitness stagnation by randomly produced chromosomes or by chromosomes produced from a series of hill climbing steps inter-mixed with diversity guided mutations. The details of DGEP are presented in the following steps.

Step 1: Generate an initial population of M chromosomes. Each chromosome, $\boldsymbol{C}$ is represented as a pair of real valued vectors, $(\boldsymbol{x}_i, \boldsymbol{\eta}_i)$, for $i = 1, \ldots, M$; $\boldsymbol{x}_i$ is the vector composed of the parameter values being optimized and $\boldsymbol{\eta}_i$ is the standard deviation vector to perform Gaussian mutations as hill climbing steps on $\boldsymbol{x}_i$, if necessary (step 7). Each $\boldsymbol{x}_i$ (and $\boldsymbol{\eta}_i$) has n components, n being the dimensionality of the problem. Each component of $\boldsymbol{x}_i$, for $i = 1, \ldots, M$, is generated uniformly at random within its domain. All the components of $\boldsymbol{\eta}_i$, for $i = 1, \ldots, M$, are initially set to some moderate value (e.g., 3), as suggested in [59]. The iteration counter, t is initialized to 1.

Step 2: Evaluate the fitness, fit_i of each chromosome, $\boldsymbol{x}_i$. Compute the selection probability, p_i for each chromosome, $\boldsymbol{x}_i$ using $p_i = fit_i / \sum_{i=1}^{N} fit_i$. This also normalizes p_i values into $[0, 1]$.

Step 3: Repeat steps 4–5 for M times. This constitutes the child population, $C(t)$ from the parent population, $P(t)$.

Step 4: Select a chromosome, $\boldsymbol{x}_i$ based on its normalized fitness value, p_i. We employed traditional roulette wheel selection scheme which ensures fitness proportional selection. However, any other fitness based selection scheme can also be adopted.

Step 5: Apply the DGM scheme to mutate the selected chromosome, $\boldsymbol{x}_i$ to produce an offspring chromosome, $\boldsymbol{x}'_j$. Details of DGM are explained later with its pseudocode (Fig. 1). Insert $\boldsymbol{x}'_j$ into the child population, $C(t)$.

Step 6: Merge the parent population, $P(t)$ and child population, $C(t)$. Sort the chromosomes according to their fitness values and select the best M chromosomes to constitute the next generation candidate population, $Q(t)$.

Step 7: Check for chromosomes trapped into deep local optima. If a particular chromosome, $\boldsymbol{x}_i$ in $Q(t)$ has not been improved for several (say, u) consecutive iterations, then allow $u/2$ DGM mutations randomly intermixed with $u/2$ hill climbing steps by Gaussian mutations using $N(0, \boldsymbol{\eta}_i)$ distribution, as in [6]. If this leads to fitness improvement, replace the original chromosome with the newly produced chromosome. Otherwise eliminate both the chromosomes from the population.

Step 8: Elimination of duplicate chromosomes. If the Euclidean distance between the genotypes of two chromosomes, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is less than a certain fraction, l of the average Euclidean distance among all the genotypes across the population, then $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are considered as duplicates. Select the one with lower fitness value and eliminate it from the population.

Step 9: If the current number of chromosomes is less than M in the candidate population, $Q(t)$ then introduce new chromosome, $\boldsymbol{x}_i$ placed uniformly at random across the search domain using: $x_{ij} = min_j + rand(0,1) * (max_j - min_j)$ for $j = 1, \ldots, n$; repeat this step until $Q(t)$ consists of M chromosomes. Now, $Q(t)$ becomes the parent population, $P(t)$ for the next iteration.

Step 10: Keep track of the best chromosome found so far. Also, increase the iteration counter, t by setting $t = t + 1$.

Step 11: Check for termination. If the best chromosome found so far is acceptable or the iteration counter, t exceeds some predefined maximum number of iterations, stop the process and return the best chromosome. Otherwise go back to step 2 and repeat again.

The essence of DGEP is the DGM scheme (step 5) for mutation. Also, DGM is supported by some simple diversity preserving schemes, as described in steps (7) and (8). Before proceeding to the details of the DGM scheme, we address a number of issues and present some implementation details in the following points.

(a) In step 1, the initial values of standard deviations, i.e., $\boldsymbol{\eta}_i$'s are all set to 3.0. This is not ad-hoc, rather only to be identical with the initial parameter settings of CEP [6], IFEP [6] and ALEP [5] for a fair performance comparison. Though such a choice of initial values does not consider the search space along each dimension, the self-adaptive mechanism for standard deviations (i.e., Eq. (2) in [6] for CEP and IFEP, Eq. (23) in [5] for ALEP) automatically adapts each $\boldsymbol{\eta}_i(j)$ separately and makes it suitable for an effective search around the corresponding individual, $\boldsymbol{x}_i$ along the dimension, j.

(b) In step 2, how to compute the fitness of an individual, $\boldsymbol{x}_i$? Since the objective function, f is to be minimized, so smaller values of $f(\boldsymbol{x}_i)$ should translate into higher fitness values. Also, both positive and negative values of $f(\boldsymbol{x}_i)$ need to be considered. In our implementation, we have employed the following formula to compute the fitness value of an individual, $\boldsymbol{x}_i$ from its objective value, $f(\boldsymbol{x}_i)$.

$$fitness(\boldsymbol{x}_i) = \begin{cases} 1/(1 + f(\boldsymbol{x}_i)) & \text{if } f(\boldsymbol{x}_i) \geq 0 \\ 1 + abs(f(\boldsymbol{x}_i)) & \text{if} |f(\boldsymbol{x}_i) < 0 \end{cases}$$

(c) Since each mutation operation by the DGM scheme (step 5) possesses both exploitative and explorative potentials, so we are quite indiscriminate to pick the selection operators (in steps 4 and 6), without expecting any special exploitative or explorative pressure from the selection operators. For parent selection, we have employed roulette wheel selection (i.e., fitness proportional selection) in step 4 and for survivor selection, we employ truncation selection (step 6). Both the selection operators are a bit exploitative in order to maintain good convergence speed. However, the choice of these selection operators is not an essential component of DGEP and any other selection operator may be used. One advantage with both these operators is that they do not require any parameter that need to be set carefully by the user using problem specific knowledge. Some other popular selection operators, e.g., the tournament selection, can also be employed. However, tournament selections require a parameter (e.g., the tournament size) which need to be set carefully and can affect the exploitative–explorative bias of the algorithm.

(d) A prolonged fitness stagnation of an individual may be caused not only by a deep local optimum, but also a wide, flat plateau of the search space. To detect such a situation, our implementation equips each individual with a counter that keeps track of how many consecutive mutation attempts fail to improve the individual. If the counter exceeds a maximum allowed value (say, u), then we assume that the chromosome is stuck either at a deep local optimum or within a wide flat plateau of the search space. So, we initiate the sequence of DGM mutations and hill climbing steps by Gaussian mutations, as mentioned in step 7 of DGEP. Our implementation of step 7 operates like this: let $\boldsymbol{x}_i$ be the individual stuck at a local optimum or a flat plateau. Then repeat the following operations on $\boldsymbol{x}_i$ for u times: Randomly

Diversity Guided Mutation (x): *Returns* $\boldsymbol{x}'$ *produced by diversity guided mutation on* $\boldsymbol{x}$

1. N_x: a set of chromosomes that have minimum distance from the chromosome, $\boldsymbol{x}$
2. Pick two chromosomes $\boldsymbol{y}$ and $\boldsymbol{z}$ uniformly at random from N_x and $P(t) - N_x$ respectively
3. Compute d_1, d_2: distance of the point, $\boldsymbol{x}$ from the points, $\boldsymbol{y}$ and $\boldsymbol{z}$ respectively
4. A: a set of genes of $\boldsymbol{x}$ picked at random for mutation
5. For each gene, j in A

   begin

   K = uniform_random~[0, 3]

   $child1.j$ = uniform_random~[$x.j - K*d_1$, $x.j+K*d_1$]

   $child2.j$ = uniform_random~[$x.j–K*d_2$, $x.j+K*d_2$]

   end
6. Apply greedy selection between *child1* and *child2* to select the fitter one. Let, **x'** be the selected offspring.
7. Return **x'**

Fig. 1. Pseudocode for DGM.

flip a fair, unbiased coin. If it comes up with head, then mutate and replace $\boldsymbol{x}_i$ by DGM mutation. Else, apply Gaussian mutation on $\boldsymbol{x}_i$ following the self-adaptive mechanism using $\boldsymbol{\eta}_i(j)$'s, in the same way as in Eqs. (1) and (2) of [6]. While the DGM mutation is always accepted, the Gaussian mutation is accepted only when it leads to fitness improvement; hence we call it hill climbing steps by Gaussian mutations. Please note again that, with each Gaussian mutation, the $\boldsymbol{\eta}_i(j)$'s (i.e., the standard deviations of Gaussian distribution, maintained separately for each individual $\boldsymbol{x}_i$ and for its each dimension, j) also go through the self-adaptation scheme, as described by Eq. (2) in [6]. Thus the sequence of perturbations attempt to break free the individual and improve its current situation, no matter whether its fitness stagnation is due to deep local optima or due to a flat plateau of the search space.

(e) Function evaluations (FEs) are often the most prohibitive and limiting section of evolutionary algorithms. In our implementation of DGEP, we employ a population of size of 50 which is half the population size of CEP and IFEP, because each DGM mutation involves two (rather than one) function evaluations for the two different mutations (i.e., exploitative and explorative). So, DGEP still uses the same number of FEs as in CEP, IFEP and ALEP. Also, with half the population size, the computational cost of average Euclidean distance across the population (required in step 8) is not supposed to be a significant burden. Our naïve implementation requires $O(M^2)$ extra computations, where M is our reduced population size (i.e., 50). However, it could be further reduced to $O(M)$ if we resort to the following reasonably approximate implementation: compute the centroid of the population ($O(M)$ computations), then compute the distance of each individual to the centroid (another $O(M)$ computations) and finally take the mean of the distances (again, $O(M)$). The neighbors of an individual, which is required by the DGM scheme (step 5 of DGEP), can also be approximated by the following procedure: sort the individuals according to their distance to the centroid ($O(M \log(M))$ computations), then for each individual $\boldsymbol{x}_i$ pick some q individuals from its adjacent left and right members in the sorted array, then select the required number of nearest individuals (i.e., $|N|$ neighbors) from these q individuals. For a constant, small value of q, the complexity still remains of $O(M \log(M))$.

(f) In every generation, DGEP updates the average Euclidean distance across the population and also updates the pairwise Euclidean distances for the individuals. With this information ready at hand, implementing step 8 requires only the value of the parameter, l. The specific value of l, along with the other parameter values of DGEP in our implementation is provided in Section 4.1.

(g) In every generation, DGEP computes the pairwise distance between individuals and uses this distance to pick the neighbors. For each individual, $\boldsymbol{x}_i$ its neighbors are selected to be the $|N|$ individuals that have the smallest Euclidean distance from it. As the individuals are continuously evolving and changing, neighbors of an individual may vary with generations, bur neighborhood size $|N|$ is kept constant all through the evolution. The value of $|N|$ in our implementation, along with other parameter values, is specified in Section 4.1.

The details of DGM scheme with its pseudocode and a brief analysis on how DGEP tries to balance between exploitations and explorations by employing population diversity information and how it achieves better immunity against premature convergence is presented the following sub-sections. Then, we make a number of points where DGEP differs from some other existing works in literature.

3.1. DGM

The central component of the proposed DGEP system is the DGM scheme for mutation. In order to mutate a particular chromosome $\boldsymbol{x}_i$, DGM randomly picks two chromosomes, one from its neighbors (i.e., most similar chromosomes) and the other from the rest of the population. Suppose d_1 and d_2 be the distance values of these two chromosomes from $\boldsymbol{x}_i$. Since d_1 is the distance between two neighbors, it is expected to be pretty small, while d_2 being the distance between two non-neighbor chromosomes across the population is expected to be rather large in comparison to d_1 (unless the entire population has converged). DGM employs d_1 to produce a small, exploitative variation on $\boldsymbol{x}_i$ while d_2 is used to bring a large, explorative variation. Based on the fitness value of both these newly produced offspring of $\boldsymbol{x}_i$, only one is selected to compete for insertion into the next generation population. In order to assist the DGM scheme, an adequate amount of diversity is sought by adopting some simple techniques, such as elimination of duplicate chromosomes by randomly produced chromosomes, replacement of chromosomes stuck at strong local optima or wide flat plateau by

multiple hill climbing steps randomly intermixed with DGM mutations, as described in steps (7) and (8) of DGEP. The details of DGM are presented in the pseudocode in Fig. 1.

DGM involves three individuals to produce an offspring. So there might be some difference of opinions on whether the proposed DGM scheme can be rightfully called a mutation operator, because it requires and recombines information from two other individuals, while a traditional mutation operator usually does not make use of any extra information from other individuals. However, as is found in the entire differential evolution (DE) literature, a DE mutation operator is also based on using information from three other individuals from the population. In a DE mutation, a direction vector is computed from one individual to another, which is then added to a third individual. The DGM scheme is quite similar to the DE mutation, so we do not find any problem to use the term 'mutation' for it which is completely consistent with the DE tradition. Apart from the similarity with the DE mutation, there is also significant difference between DGM and DE. Usually DE is centered around distances in gene level, while DGEP is based on distances among chromosome (i.e., individual) level. While the DE mutation does not show any concern for exploitation or exploration and maintains no relation with the existing population diversity, DGM puts its active effort to balance exploitation with exploration by estimating the existing population diversity across similar and dissimilar individuals and employing the information for exploitative and explorative mutations. These two different mutations with widely different step sizes are actually merged to constitute the DGM mutation scheme. DGM produces much improved results than the recent DE variants, like NSDE [27,28], as becomes apparent from the experimental study (e.g., Table 9).

For every benchmark function that has been employed, the search spaces along all the dimensions (i.e., the domains of values for all the genes) are equal and identical. So, the adjustment of the genes by DGM uses the same amplitude of step size for all the genes. This may not be a good approach in many real world problems where the domains of the values corresponding to different genes are widely different. In practical applications it is possible to have one design variable in [0, 1] while another one within [0, 1000]. However, such a situation can be easily handled by a simple linear scaling of all the domains to the same size. Before applying the DGM mutation, every gene value, x_i with domain $[min_i, max_i]$ can be preprocessed by a simple linear transformation function, g that transforms each dimension to [0, 1] by

$$g(x_i) = \frac{(x_i - min_i)}{(max_i - min_i)}$$

In the pseudocode of DGM (Fig. 1), K operates as a random scaling factor of the distance between individuals in the DGM mutation scheme. As we have found with some experimentations, a random value of K within some small range (e.g., uniform_random ~ [0, 3] as implemented in DGM) often provides better results than a fixed or larger value of K. The primary objective of exploration and exploitation is conducted by the distances among neighbors and non-neighbors (i.e., d_1 and d_2 in the pseudocode), not by K. The role of K is essentially a bit of further improvements of the results, as we have been found with some experimental study.

3.2. Diversity guided explorations and exploitations

In this section we present how DGM tries to balance between explorations and exploitation using population diversity information. Mutations with small step size usually induce small amount of variations to the parent chromosomes and thus exploits existing solutions until the population gets stuck at local optima. In contrast, mutations with large step size are more likely to escape from local optima and thus better suited for search space explorations, but may lack the stability to perform in-depth tuning of the existing solutions and thus may fail to locate the global optima. Some existing works try to control mutation step size following some rigid adaptation strategy [31,32] or using different distributions [5,6] or outcomes of previous mutations [38]. In contrast, DGM adapts the mutation step size by picking distance values from the population. Exploitation is conducted by picking two very similar chromosomes and employing the small genotype distance between them which ensures a small step size that is suitable for exploitation. If this distance value is d_1, the mutation step size is generated uniformly at random from $[0, K^* d_1]$. Exploration is carried out following the same way, but the distance between two non-neighbors, say d_2, is employed and the mutation step size is generated uniformly at random from $[0, K^* d_2]$. The pair of distance values, d_1 and d_2 acts as sampled, approximate diversity of neighbors and non-neighbors across the population. Thus the distribution and distances of neighbors and non-neighbors from a chromosome controls the exploitations and explorations around the chromosome. During the early generation both the neighbors and non-neighbors would be far apart from each other. So the distance values, d_1 and d_2 are likely to be large in most instances and the evolutionary search would be mostly explorative. As the population evolves, chromosomes would approach the local peaks with the formation of several neighbor groups within the population. However, considering a high dimensional multimodal problem with number of local optima much higher than the number of evolving chromosomes (which is usually the case), the distance between different neighborhoods would still be significantly large. So, inter neighborhood diversity, d_2 would be sufficiently high to carry out explorations in parallel to the exploitations with small intra-neighborhood diversity, d_1. Unless the entire population converges to a single global optimum, both explorations and exploitations would continue and guide the evolution to better solutions. In addition to the mutation operator, the selection operator of DGEP also possesses both explorative and exploitative features. While the fitness based selection exerts mainly exploitative pressure on the evolution, the elimination of duplicate chromosomes and other chromosomes that get trapped at local optima (i.e., no fitness improvement for a long time) puts explorative pressure on the evolutionary search. The synergy of these explorative and exploitative forces ensures better balance and higher amount of population diversity in comparison to classical evolutionary approaches, as becomes apparent from the experimental results in the next section (e.g., Table 3 and Fig. 5).

Balancing exploitation and exploration has also been addressed by some other works, such as RCMA [25], NSDE [27,28] and LSRCMA [26]. All these works differentiate the degree of exploitations and exploration on different chromosomes based on their fitness values. This probably makes them more exploitative than explorative. DGEP does not differentiate the degree of exploitations and explorations on the chromosomes; rather it attempts both explorative and exploitative variation from each chromosome. Also, unlike RCMA [25], NSDE [27,28] and LSRCMA [26], the selection operator of DGEP puts some explorative pressure with elimination of duplicate and stagnant chromosomes. Thus both the evolutionary operators of DGEP, i.e. selection and mutation, exhibit awareness towards population diversity to avoid premature convergence.

To some extent, the degree of exploration that DGEP is able to conduct might be controlled by K, as found in the pseudocode (step 5). K acts as a scaling factor of the distance, say d, between neighbors or non-neighbors. For example, consider an optimization problem with only one parameter. A chromosome with a parameter value v is mutated uniformly at random within the range $[v - K^* d, v + K^* d]$. The standard deviation of the mutation perturbation would be $K * d/\sqrt{3}$, which is proportional to both K and d. So, larger values of K (and d) are likely to provide better explorations. However, more exploration is not always helpful, especially

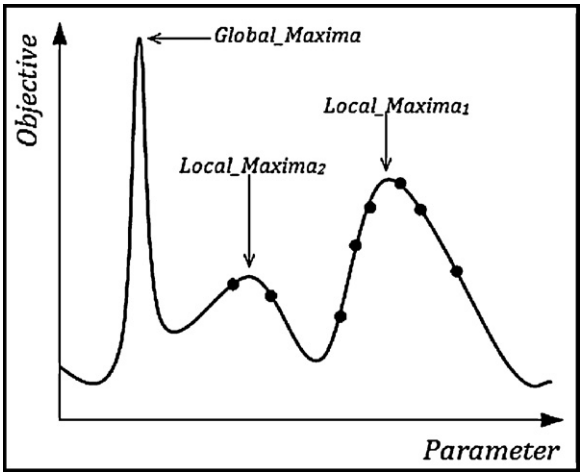

Fig. 2. Scenario #1 – the entire population has converged to a few local optima missing the narrow global optimum.

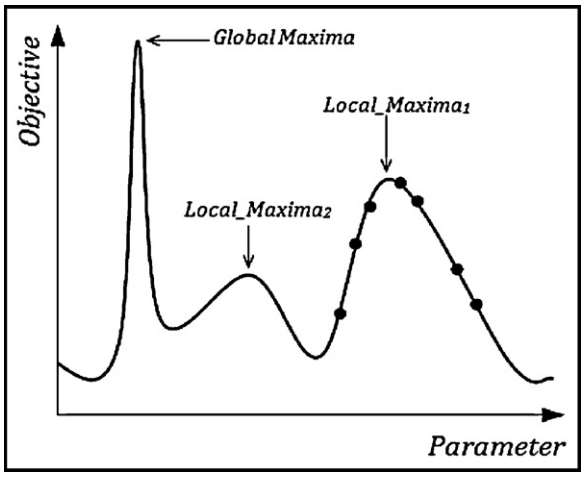

Fig. 3. Scenario #2 – the entire population has converged to a single local optimum far from the narrow global optimum.

once the neighborhood of the global optima is found or faster convergence is sought. Instead, what is always essential is a proper balance between explorations and exploitations. With our scheme of choosing d from two different ranges of values (exploitative and explorative), we leave up K to a simple random value. With some experiments, it is observed that a uniform random value of K within some small range, e.g., [0, 3], produces sufficiently good results and is often better than a fixed value of K. So, K is simply picked uniformly at random from [0, 3]. This is quite arbitrary and meant not to be optimal. With such simple choice of K, exploitation and exploration is mainly carried out by choosing between d_1 and d_2, i.e., the exploitative and explorative distances to guide the mutation step size.

3.3. Avoiding premature convergence

To avoid premature convergence, the search operators should have some capacity to escape from local optima. In the following two scenarios, we depict how DGEP possesses an inherent immunity against local optima and can conduct search until the population is driven to the global optima. For the ease of visualization, a maximization problem with only one parameter is considered. Scenario #1 (Fig. 2) depicts a situation where the search process has failed to spot the global maxima and the entire population has converged to two local maxima. In this situation, the average distance among neighbors would be pretty small and help the chromosomes hill climb towards locally optimal fitness values, $Local\_Maxima_1$ and $Local\_Maxima_2$, as depicted in Fig. 2. If the different peaks are well-separated with significant amount of distance between them, it would be quite difficult for traditional mutation operators to have sufficiently large step size in order to escape from the local maxima. However, the higher the distance between the two neighborhoods, the more explorative mutation step size DGEP would pick from their distances which makes it easier for DGEP to break free from the local maxima. Once the vicinity of the global maximum is found, both the intra-neighborhood and inter-neighborhood distances start to drop and allow DGEP to converge to the global maximum.

The worst possible scenario that can appear during evolutionary search is the scenario #2 (Fig. 3) where the entire population has converged to a single local maximum far from the narrow global maximum. If the distance to *Global_Maxima* is significantly large, it would be extremely difficult to improve the scenario by using any simple, e.g. [39] or adaptive, e.g. [35,36], mutation scheme. Use of specialized selection operator, e.g. [49,60,61] or probability distributions, e.g. [5,6] for mutation would not help in such a situation. This is because the appropriately large mutation step size in such situations cannot be derived from the population information and the traditional fitness based selection operator discards the newly produced offspring that may be actually closer to the *Global_Maxima* but have lower fitness value. However, DGEP still possess some probability to improve the situation because of its specialized selection operator. As the entire population climbs towards the single local maximum that entraps the entire population (i.e., $Local\_Maxima_1$), some chromosomes would be more or less duplicates of each other. Also, some chromosomes, after reaching the fitness of $Local\_Maxima_1$ (Fig. 3) would get into fitness stagnation for a long time. This triggers the selection operator, as described in steps (7) and (8) of DGEP, to replace these chromosomes by randomly producing chromosomes across the search space and applying a sequence of diversity guided mutations intermixed with hill climbing mutation steps before exerting selection pressure on them. Random placement of the offspring chromosomes usually ensures large variations from the current points while the sequence of hill climbing and mutation steps provide them with good survival probability when compared to the existing chromosomes. As a result, some of the new chromosomes may reach some other local maxima and turn this scenario into the previous scenario #1 (i.e., the population contains multiple local maxima) which is much easier to deal with. Thus DGEP ensures better probability that the search for global optima continues avoiding permanent premature convergence around local optima.

3.4. Differences with existing works

DGEP differs from most other existing approaches in a number of ways. First, it neither tries to find an optimal mutation rate and step size (e.g. [39]) nor tries to adapt their values using some adaptation rule (e.g. [35,36]). It rather uses existing distances among chromosomes as a measure of diversity across neighbors and non-neighbors of the population and guides the mutation operation with this information. Mutation step size is not explicitly controlled by any rigid adaptation formula, rather becomes guided by the distances between chromosomes within the search space. As the fitness landscape may have complex and highly non-linear characteristics, any fixed adaptation rule based on generation number (e.g., similar to ES schemes [35,36]) is likely to fail to respond to the continuously evolving optimization scenario.

Second, DGEP tries to balance the exploitative and explorative search requirements during each mutation by producing two

Table 1
Benchmark functions used in the experimental study. D is the dimensionality of the function, S is the domain of the variables, f_{min} is the function value at global minima, and C is function characteristics with values U: unimodal, M: multimodal, S: separable, N: non-separable.

No	Function	S	D	C	f_{min}	Formulation
f_1	Sphere	$[-100, 100]$	30	US	0	$f_1(x) = \sum_{i=1}^{n} x_i^2$
f_2	Schwefel 2.22	$[-10, 10]$	30	UN	0	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
f_3	Schwefel 1.2	$[-100, 100]$	30	UN	0	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$
f_4	Schwefel 2.21	$[-10, 10]$	30	US	0	$f_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i < n\}$
f_5	Rosenbrock	$[-30, 30]$	30	UN	0	$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$
f_6	Step	$[-100, 100]$	30	US	0	$f_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$
f_7	Quartic	$[-1.28, 1.28]$	30	US	0	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$
f_8	Schwefel	$[-500, 500]$	30	MS	-12569.5	$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
f_9	Rastrigin	$[-5.12, 5.12]$	30	MS	0	$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$
f_{10}	Ackley	$[-32, 32]$	30	MN	0	$f_{10}(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$
f_{11}	Griewank	$[-600, 600]$	30	MN	0	$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
f_{12}	Penalized	$[-50, 50]$	30	MN	0	$f_{12}(x) = \frac{\pi}{n}[10\sin^2(\pi y_1) + (y_n - 1)^2] + \sum_{i=1}^{n} u(x_i, 10, 100, 4) + \frac{\pi}{n}\left[\sum_{i=1}^{n-1} (y_i - 1)^2 (1 + 10\sin^2(\pi y_{i+1}))\right], \quad y_i = 1 + \frac{1}{4}(1 + x_i)\psi$
f_{13}	Penalized2	$[-50, 50]$	30	MN	0	$f_{13}(x) = 0.1\,[\sin^2(\pi x_1) + (x_n - 1)^2(1 + \sin^2(2\pi x_n))] + \sum_{i=1}^{n} u(x_i, 5, 100, 4) + 0.1\left[\sum_{i=1}^{n-1} (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1}))\right]$
f_{14}	Foxholes	$[-65.536, 65.536]$	2	MS	1	$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}\right]^{-1}$
f_{15}	Kowalik	$[-5, 5]$	4	MN	3.07×10^{-4}	$f_{15}(x) = \sum_{i=1}^{11} \left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$
f_{16}	Six Hump Camel Back	$[-5, 5]$	2	MN	-1.0316	$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
f_{17}	Branin	$[-5, 10]$ $x[0, 15]$	2	MS	0.398	$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$
f_{18}	GoldStein-Price	$[-2, 2]$	2	MN	3	$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
f_{19}	Hartman3	$[0, 1]$	3	MN	-3.86	$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right]$
f_{20}	Hartman6	$[0, 1]$	6	MN	-3.32	$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$
f_{21}	Shekel5	$[0, 10]$	4	MN	-10	$f_{21}(x) = -\sum_{i=1}^{5} \left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$
f_{22}	Shekel7	$[0, 10]$	4	MN	-10	$f_{22}(x) = -\sum_{i=1}^{7} \left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$
f_{23}	Shekel10	$[0, 10]$	4	MN	-10	$f_{23}(x) = -\sum_{i=1}^{10} \left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$

different offspring from each chromosome: one with small step size to facilitate exploitation around the parent chromosome and the other with large enough step size for search space exploration. Most existing approaches do not use such explicit measure to balance between exploitations and explorations. The use of two widely different ranges of mutation variations provides DGEP an inherent immunity against local minima.

Third, most existing approaches (e.g. [53–61]) alter either the selection operator or the variation operator to promote diversity and prevent premature convergence, while DGEP alters both its variation and selection operators. While the DGM mutation scheme plays the key role of employing diversity information for effective mutations, the selection operator also assists it by promoting diversity with the elimination of the chromosomes that are either duplicates or strongly trapped in local optima. Altering only one operator (e.g., selection) and leaving the other (i.e., mutation) unchanged may cause the operators to play reverse roles in the evolution and nullify each other's effect resulting in unacceptably slow convergence speed.

Fourth, some approaches, e.g. [42,44,53] attempt to estimate the population diversity using some diversity metric and try to make key decisions based on this diversity value. Since there is no generally accepted metric to measure population diversity, DGEP uses very general all-purpose Euclidean distance to measure the distances between the genotypes of the chromosomes. Using simple metric provides a way for plain interpretation and analysis of the operations.

Table 2
Comparison among DGEP, CEP [2], IFEP [6] and ALEP [5] on the classical benchmark functions. Results are averaged over 50 independent runs. The best result for each function is marked with boldface font.

No	f_{min}	DGEP		CEP		IFEP		ALEP	
		Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
f_1	0	$\mathbf{5.42 \times 10^{-8}}$	1.48×10^{-8}	2.2×10^{-4}	5.9×10^{-4}	4.16×10^{-5}	–	6.32×10^{-4}	6.4×10^{-5}
f_2	0	$\mathbf{6.64 \times 10^{-12}}$	3.61×10^{-12}	2.6×10^{-3}	1.7×10^{-4}	2.44×10^{-2}	–	–	–
f_3	0	$\mathbf{4.13 \times 10^{-8}}$	9.25×10^{-9}	5.0×10^{-2}	6.6×10^{-2}	–	–	4.18×10^{-2}	5.97×10^{-2}
f_4	0	**1.06**	0.32	2.0	1.2	–	–	–	–
f_5	0	**1.28**	0.76	6.17	13.61	–	–	43.40	31.52
f_6	0	**0**	0	577.76	1125.76	–	–	–	–
f_7	0	$\mathbf{1.94 \times 10^{-12}}$	8.23×10^{-13}	1.8×10^{-2}	6.4×10^{-3}	–	–	–	–
f_8	−12569.5	**−12567.4**	0.18	−7917.1	634.5	–	–	−11469.2	58.2
f_9	0	$\mathbf{1.56 \times 10^{-12}}$	5.12×10^{-9}	89.0	23.1	–	–	5.85	2.07
f_{10}	0	$\mathbf{2.47 \times 10^{-16}}$	6.83×10^{-17}	9.2	2.8	4.83×10^{-3}	–	1.9×10^{-2}	1.0×10^{-3}
f_{11}	0	$\mathbf{7.52 \times 10^{-14}}$	2.54×10^{-14}	8.6×10^{-2}	0.12	4.54×10^{-2}	–	2.4×10^{-2}	2.8×10^{-2}
f_{12}	0	$\mathbf{3.32 \times 10^{-12}}$	1.94×10^{-13}	1.76	2.4	–	–	6.0×10^{-6}	1.0×10^{-6}
f_{13}	0	1.74×10^{-4}	2.65×10^{-5}	1.4	3.7	–	–	$\mathbf{9.8 \times 10^{-5}}$	1.2×10^{-5}
f_{14}	1	**1.02**	0.04	1.66	1.19	–	–	–	–
f_{15}	3.07×10^{-4}	$\mathbf{2.17 \times 10^{-4}}$	8.2×10^{-5}	4.7×10^{-4}	3.0×10^{-4}	–	–	–	–
f_{16}	−1.0316	**−1.031**	0.00	**−1.031**	4.9×10^{-7}	–	–	**−1.031**	0.00
f_{17}	0.398	**0.398**	2.4×10^{-7}	**0.398**	1.5×10^{-7}	–	–	–	–
f_{18}	3	**3.000**	0.000	**3.0**	0	–	–	**3.000**	0.000
f_{19}	−3.86	**−3.86**	9.4×10^{-3}	**−3.86**	1.4×10^{-2}	–	–	–	–
f_{20}	−3.32	**−3.32**	2.5×10^{-4}	−3.28	5.8×10^{-2}	–	–	–	–
f_{21}	−10.57	**−9.87**	0.54	−6.86	2.67	−6.46	–	−9.54	1.69
f_{22}	−10.57	**−10.47**	0.08	−8.27	2.95	−7.10	–	−10.30	0.74
f_{23}	−10.57	-10.51	3.8×10^{-2}	−9.10	2.92	−7.80	–	**−10.54**	4.9×10^{-5}

Fifth, unlike evolutionary systems that employ complex procedures to promote population diversity, e.g., island [17], nation [19], religion [20], immigration [62], reserve population [44], switching to and from different genetic operations [42], DGEP employs very simple diversity preserving measures. DGEP eliminates only those chromosomes that do not contribute significantly in the evolutionary search, i.e., chromosomes that are duplicates or almost identical to some other existing chromosome and the chromosomes that have not been improved for a long time. However, in the latter case, DGEP makes a series of variation operations to improve it before discarding it completely. Simple measures for diversity preservation make the system open for clear interpretation and easier analysis.

4. Experimental studies

This section presents the empirical evidence for the better effectiveness of the proposed DGEP scheme over a number of existing evolutionary systems. Many evolutionary systems exist in the literature against which we could compare DGEP. However, since DGEP uses mutation as the sole variation operator and makes use of population diversity information to adjust the mutation variation and to balance between exploitations and explorations, we primarily consider Classical EP (CEP) [2], Improved Fast EP (IFEP) [6], Adaptive EP with Lévy Mutation (ALEP) [5], Island Model GA (IMGA) [17,18], Restricted Truncation Selection (RTS) [52], Dual Population Genetic Algorithm (DPGA) [44], Real Coded Memetic Algorithm (RCMA) with Crossover Hill Climbing (XHC) [25], Comprehensive Learning Particle Swarm Optimizer (CLPSO) [63,64], RCMA with Adaptive Local Search (LSRCMA) [26], Differential Evolution with Neighborhood Search (NSDE) [27,28] and Covariance Matrix Adaptation Evolution Strategy (CMAES) [38] for comparison. Like DGEP, both IFEP [6] and ALEP [5] use only mutation for producing offspring. IFEP [6] has mixed, rather than switched, Cauchy and Gaussian mutations in one algorithm. This algorithm generates two candidate offspring from each parent: one by Cauchy mutation and one by Gaussian mutation. The better candidate is then chosen by IFEP [6] as the offspring. ALEP [5], on the other hand, generates four candidate offspring from each parent by Lévy mutation with four different distributions. It has been shown that IFEP [6] and ALEP [5] perform better than either their non-adaptive versions or the classical EP (CEP) [2]. IMGA [17,18] uses multiple sub-populations following the island model [17] with periodic exchange of individuals between the islands according to some predetermined migration policy. The isolated nature of the islands helps the populations evolve separately which lowers gene flow and promotes population diversity. RTS [52] is a crowding method that is somewhat different from the standard crowding and usually produces better results than all other variants of crowding. DPGA [44] maintains a reserve population in addition to the main population for promoting diversity. Both populations evolve through generations, but with completely different objectives. While the main population evolves to optimize fitness value, the reserve population evolves with the purpose of providing useful diversity around the best chromosomes of the main population. The distance value between the two populations is mostly adapted for exploitations, but switched to explorations when the main population gets trapped into local optima for several generations. RCMA with XHC [25] executes explorative and exploitative operations separately and combines them in one algorithm. It uses PBX crossover [25] and BGA mutation [65] for exploration. RCMA employs a negative assortative mating strategy for selecting two parents to perform crossover in order to introduce population diversity. RCMA with a specialized crossover operator, XHC [25] has been shown to perform better than all other variants. NSDE [27,28] and LSRCMA [26]

tries to balance between exploitations and explorations by combining the benefits of exploitative local search or neighborhood search with their more explorative component of differential evolution or memetic algorithm, respectively.

4.1. DGEP on classical benchmark functions

A set of 23 classical benchmark test functions [6,63] was used in the experiments. Based on their properties, the functions can be divided into three groups: functions with no local minima, many local minima, and a few local minima. The analytical form of these functions are given in Table 1, where D denotes the dimensionality of the problem, S is the search range of the variables, C is the function characteristics and f_{min} is the function value at the global minimum. The first seven functions, f_1–f_7 are high dimensional unimodal functions. Functions f_8–f_{13} are high dimensional multimodal functions with many local minima and highly non-linear characteristics. They have hundreds of local optima, even when the dimension is just two. The number of local optima increases exponentially as the dimension increases. Their complex nonlinear characteristics combined with exponential many local optima make them extremely difficult to optimize. For example, Ackley's function, f_{10} exhibits one narrow global optimum basin and exponentially many minor local optima. Griewank's function f_{11} has a component causing linkages among variables, thereby making it extremely difficult to reach the global optimum by mutating any subset of the variables. The complexity of Schwefel's function, f_8 is due to the deep local optima being far from the global optimum. The remaining functions, f_{14}–f_{23} are low-dimensional multimodal functions with a small number of local minima. A more detailed description of each function can be found in [6,63]. In all the following experiments, the population size was set to 50. The number of function evaluations (FEs) was set to 150,000 for functions f_1–f_{13} and 10,000 for the low dimensional functions, f_{14}–f_{23}. These values are chosen to make a fair comparison with previous works. The remaining parameters of DGEP are u (maximum number of generations without fitness improvement before discarding a chromosome), l (ratio of genotype distances to determine duplicates), $|N|$ (number of neighbors for each chromosome) and $|A|$ (number of genes to be mutated by the DGM scheme). Except with some extreme choice of values, the performance of DGEP is not very sensitive to these parameters. We have employed DGEP with $u = 50$, $l = 0.01$, $|N| = 5$ and $|A| = 1$. These values are quite arbitrary and meant not for optimum. According to [66], CEP has been implemented for the same population size and FEs.

Table 2 shows the mean best results of DGEP on 23 classical test functions along with CEP [2], IFEP [6] and ALEP [5]. The results of ALEP and IFEP are presented only for 14 and 7 functions, as available in [5] and [6], respectively. With no more results of ALEP and IFEP on the remaining functions, we cannot compare the DGEP with them on those functions. These missing results are shown with '–' in Table 2. However, as DGEP outperforms both ALEP and IFEP on almost all the available functions, we can assume similar performance on the remaining functions, too. All the results have been averaged over 50 independent runs. Fig. 4 shows the convergence characteristics of DGEP and CEP [2] on several functions in terms of the mean best fitness value. It is evident from the results that DGEP arrived sufficiently close to the global minima for most of the functions and accomplished this with high degree of consistency, as evident from the low standard deviation values. Many EP and ES schemes fail to do this [31], especially for the multimodal functions f_8–f_{23}. It is seen from Fig. 4 that DGEP achieved nearly log-linear convergence until it reached sufficiently close to the global minima. The evolutionary process progresses smoothly, almost no where getting stuck at local minima until it reaches very close proximity of the global minima, while many evolutionary algorithms expend significant amount of time being stuck at several intermediate local minima [31]. As obvious from Table 2, the performance of DGEP is remarkably better than the other three algorithms. The t-test (not shown in the table) confirms with 95% certainty that the improvement is statistically significant for most of the functions. DGEP is significantly better than IFEP [6] on seven out of the seven functions. Also, DGEP has outperformed ALEP [5] on 10 out of 14 functions, shows equal performance on two functions while ALEP [5] performs better on only the remaining two. In comparison to CEP [2], DGEP outperforms it on 19 out of 23 functions while on the remaining four functions both of them show similar performance.

Table 3
Comparison of DGEP and CEP [2] on the population diversity for the functions, f_8–f_{13}. Diversity is measured as the average Euclidean distance of the chromosomes of the last generation. The best result for each function is marked with boldface font.

Function	CEP	DGEP
f_8	6.43×10^{-3}	$\mathbf{1.09 \times 10^{-2}}$
f_9	1.39×10^{-8}	$\mathbf{2.27 \times 10^{-3}}$
f_{10}	8.53×10^{-8}	$\mathbf{7.90 \times 10^{-5}}$
f_{11}	6.47×10^{-6}	$\mathbf{8.62 \times 10^{-3}}$
f_{12}	2.86×10^{-9}	$\mathbf{4.57 \times 10^{-4}}$
f_{13}	1.07×10^{-7}	$\mathbf{5.33 \times 10^{-4}}$

However, only three functions, f_4, f_5 and f_{21} seem to present difficulty, more or less, to all the EP algorithms in our study. They are the *Schwefel* 2.21, *Generalized Rosenbrock* and *Shekel*5 functions respectively. Although generalized Rosenbrock function has been regarded as a unimodal function, there is evidence [69] suggesting that it contains several minima in high dimensional instances. The global optimum resides inside a long, narrow, parabolic shaped flat valley. Finding the valley is not difficult, but pinpointing the global optimum in an almost flat region is extremely difficult. Both the remaining two functions, i.e., *Schwefel* 2.21 and *Shekel*5 functions, exhibit predominant flat and semi-flat search regions with narrow global basin. Searching for a narrow global basin in a flat search space with no useful gradient direction essentially turns into a search for a needle in a haystack. So, all the schemes find difficulty to locate the global optimum and often prematurely converge to some local optima, as found from the results in Table 2.

Now we investigate how the population diversity evolves under the influence of the proposed DGEP scheme compared to the simple CEP [2] scheme. While CEP [2] shows no concern to population diversity, DGEP is based on inter-neighborhood and intra-neighborhood diversities to guide its mutation. It also alters the selection operator to promote more diversity. Maintaining population diversity and avoiding premature convergence is most troublesome for the high dimensional multimodal functions, f_8–f_{13}. For these functions, Table 3 compares the last generation population diversity of CEP [2] and DGEP. Both CEP [2] and DGEP used real valued representation for the chromosomes (i.e., points in the n-dimensional parameter space), so the average Euclidean distance among the chromosomes across the population is used to measure the population diversity. Using such simple metrics to quantify diversity not only has the benefit of plain interpretation, but also is consistent with the scheme of DGEP that uses the Euclidean distance between neighbors and non-neighbors to guide mutation step size and also to find duplicate chromosomes for elimination. From the diversity present at last generations, as observed in Table 3, it is obvious that DGEP fosters significantly higher amount of diversity in comparison to CEP [2]. Fig. 5 shows the evolution of population diversity with generations for the functions f_9 and f_{13}. It is seen that CEP [2] drops diversity rapidly within a few early generations and then never becomes able to restore the lost diversity. In contrast, DGEP maintains higher amount of diversity all through the evolution and exhibits both ups and downs in the diversity

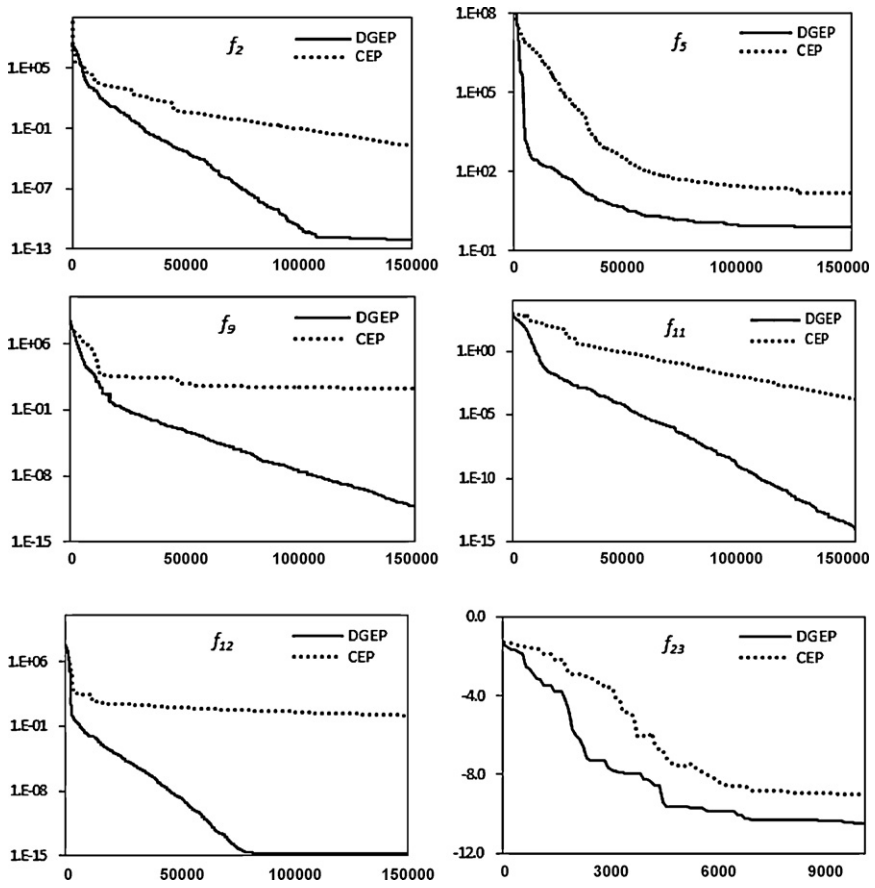

Fig. 4. Convergence characteristics of DGEP and CEP [2] on unimodal functions f_2, f_5 and multi-modal functions f_9, f_{11}, f_{12} and f_{23}. The vertical axis is the function value and the horizontal axis is the number of fitness evaluations.

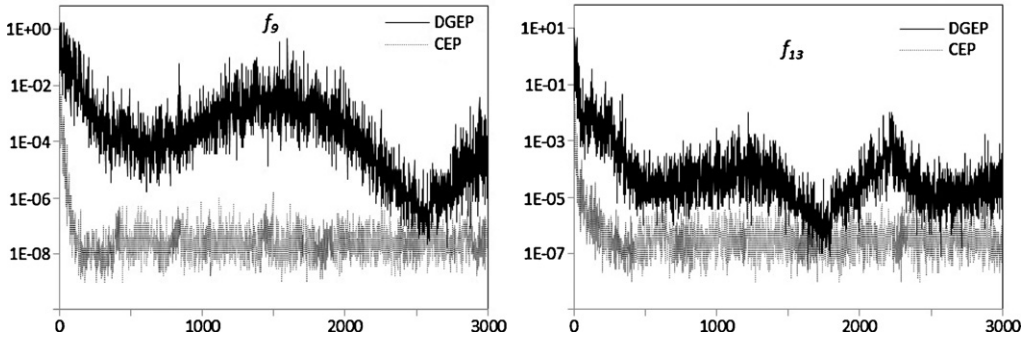

Fig. 5. Evolution of population diversity with generations under the influence of CEP [2] and DGEP for the functions f_9 (left) and f_{13} (right).

values which indicates its strength to restore significant portion of the lost diversity with its specially designed mutation and selection operators.

Table 4 compares DGEP against another algorithm, RCMA with XHC [25]. Both DGEP and RCMA [25] are applied to five functions for 100,000 FEs and with dimensionality = 25, as suggested in [25]. Table 4 shows RCMA with XHC [25] outperforms DGEP on only one unimodal function f_1, while DGEP outperforms RCMA [25] on the unimodal function f_3 and all the multimodal functions f_5, f_9 and f_{11}. Although we cannot perform t-test for lack of details of the available results, it is apparent from the results that the performance difference is statistically significant for all the functions.

The proposed system, DGEP is also compared with IMGA [17,18], RTS [52] and DPGA [44] on the multimodal functions, f_8–f_{23}

Table 4
Comparison between DGEP and RCMA with XHC [25] on five classical benchmark functions with dimension = 25. All results have been averaged over 50 independent runs.

Function	DGEP	RCMA	Function	DGEP	RCMA
f_1	1.26×10^{-11}	$\mathbf{6.5 \times 10^{-101}}$	f_9	$\mathbf{1.17 \times 10^{-14}}$	1.4
f_3	$\mathbf{1.05 \times 10^{-15}}$	3.8×10^{-7}	f_{11}	$\mathbf{9.23 \times 10^{-15}}$	1.3×10^{-2}
f_5	$\mathbf{1.83 \times 10^{-1}}$	2.2			

Table 5
Comparison among IMGA [17,18], RTS [52], DPGA [44] and DGEP on high dimensional multimodal functions, f_8–f_{13}. Results have been averaged over 50 independent runs. Best results are marked with boldface fonts.

Function	IMGA	RTS	DPGA	DGEP
f_8	−12008.1	−12443.9	−12550.5	**−12567.4**
f_9	0.358	**0**	**0**	1.56×10^{-12}
f_{10}	4.69×10^{-15}	5.26×10^{-15}	3.55×10^{-15}	$\mathbf{2.47 \times 10^{-16}}$
f_{11}	3.54×10^{-3}	2.07×10^{-3}	1.28×10^{-3}	$\mathbf{7.52 \times 10^{-14}}$
f_{12}	2.48×10^{-7}	1.57×10^{-32}	$\mathbf{1.57 \times 10^{-32}}$	3.32×10^{-12}
f_{13}	5.97×10^{-29}	2.20×10^{-4}	$\mathbf{1.39 \times 10^{-32}}$	1.74×10^{-4}

Table 6
Comparison among IMGA [17,18], RTS [52], DPGA [44] and DGEP on low dimensional multimodal functions, f_{14}–f_{23}. Results have been averaged over 50 independent runs. Best results are marked with boldface fonts.

Function	IMGA	RTS	DPGA	DGEP
f_{14}	1.410	1.018	1.355	**1.02**
f_{15}	6.53×10^{-4}	5.81×10^{-4}	5.86×10^{-4}	$\mathbf{2.17 \times 10^{-4}}$
f_{16}	−1.032	−1.032	−1.031	−1.032
f_{17}	0.398	0.398	0.398	0.398
f_{18}	3.00	3.00	3.00	3.00
f_{19}	−3.86	−3.86	−3.86	−3.86
f_{20}	−3.31	−3.28	−3.28	**−3.32**
f_{21}	−7.11	−9.14	−6.86	**−9.87**
f_{22}	−9.35	−10.04	−8.27	**−10.47**
f_{23}	−9.27	−10.24	−9.10	**−10.51**

(Tables 5 and 6). The first six functions, f_8–f_{13} are high dimensional functions for which DGEP outperforms IMGA [17,18] and RTS [52] on five and four functions respectively while they perform better only on the remaining one or two. The remaining algorithm, DPGA [44] performs better on three functions, while DGEP outperforms it on the remaining three. For the low dimensional multimodal functions f_{14}–f_{23}, DGEP always either outperforms all of IMGA [17,18], RTS [52] and DPGA [44] (on six functions) or shows similar performance (on the remaining four), as exhibited in Table 6.

Although DGEP remarkably outperforms all of IMGA [17,18], RTS [52] and DPGA [44] on the functions f_{14}–f_{23}, it receives an intense competition from DPGA [44] on the functions, f_8–f_{13}. The key difference between these two families of functions provides us an important insight. Functions f_8–f_{13} have hundreds of local optima, even with just one or two dimensions, as illustrated in Fig. 6 which shows the 2-D version of the generalized Rastrigin function, f_9. The number of local optima increases exponentially with the number of dimensions. So with the high dimensionality of 30, the fitness landscape is occupied with exponentially large number of local

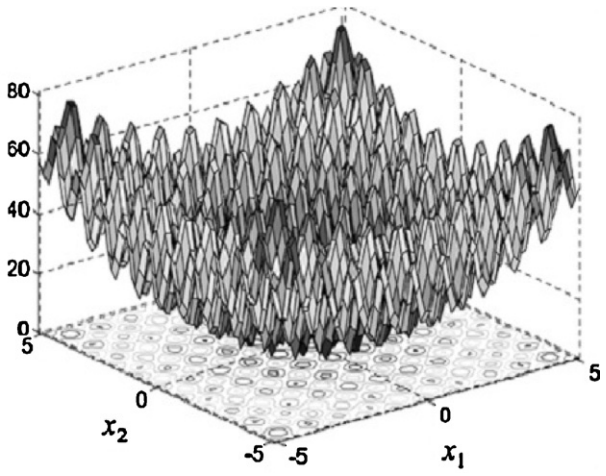

Fig. 6. Graph of 2-dimensional generalized Rastrigin's function.

minima. As a result, the distances among neighboring local peaks are usually quite small and escaping from a locally optimum point does not demand too high exploration capacity from the search operators. On the other hand, functions f_{14}–f_{23} have quite a small number (only 2–10) of local optima which makes the peaks mostly well separated. So, once the entire population converges to one or more isolated local optima, it requires relatively strong exploration capacity to break free from a locally optimum point. Both DPGA [44] and IMGA [17,18] maintain only two sub-populations, so it becomes difficult for them to represent all the distant optima. RTS [52] performs relatively better on these functions, because its selection mechanism avoids crowding of chromosomes within the same optima and fosters well separated diverse chromosomes that evolve to the different optima in parallel. Since the number of optima for these functions is usually smaller than the population size, RTS [52] can keep track of all the optima and thus shows overall better performance than both of IMGA [17,18] and DPGA [44]. DGEP performs remarkably well on f_{14}–f_{23} outperforming all of IMGA [17,18], RTS [52] and DPGA [44] because both its mutation and selection operators are well suited for such optimization scenario. The well separated local optima ensure a sufficiently large step size for the DGM scheme for mutation, while its selection operator discards similar chromosomes and fosters diverse ones that evolve towards different local optima until the global optima is found.

For the high dimensional multimodal functions f_9, f_{12} and f_{13}, DPGA [44] shows excellent strength in fine-tuning by minimizing to as low as 1e−32 which is better than the proposed DGEP. Although DGEP locates the global optima for all these functions, it cannot perform such exploitations and fine tuning of the solutions. Once DGEP locates the global optima, the entire population reaches its close proximity and all the chromosomes become somewhat similar to each other. As a result, the selection operator of DGEP eliminates the nearly duplicate chromosomes and replaces them with randomly placed chromosomes to ensure more diversity and better exploration capacity. In Fig. 5, the gradual regain of population diversity by DGEP during the late generations indicates its excellent ability of maintaining diversity, which is the key for better explorations. This strength makes it possible for DGEP to locate the global optima for all the multimodal functions, f_8–f_{23}, while all three of IMGA [17,18], RTS [52] and DPGA [44] miss the global optima on several occasions, especially for the functions f_{21}–f_{23}. However, this very strength of locating global optima by more explorations and more diversity does not allow DGEP to be completely exploitative, which is the only reason why its performance becomes worse than the extensive fine tuning of DPGA [44] on the three multimodal functions: f_9, f_{12} and f_{13}. However, a simple fine tuning scheme after the execution of DGEP may further improve its results.

In order to gain a deeper understanding of the performance difference between DGEP and CEP [2], we now compare the achievement of successful mutation rates by Gaussian mutation of CEP [2] and the proposed DGM mutation scheme of DGEP during the course of the evolution. Here success rate of a mutation is determined by the percentage of better offspring produced by that scheme. Results from Table 7 and the graphs in Fig. 7 show that the percentage of successful mutations is much higher by the DGM scheme all throughout the evolution than the traditional Gaussian mutation scheme employed by CEP [2].

The previous comparisons suggest that DGEP is better than its counterparts that use only mutation or makes use of population diversity information and/or attempts to balance explorative and exploitative operations separately. It is interesting to investigate the performance of DGEP with an approach that neither uses mutation nor considers the exploration and exploitation operations explicitly. One such approach is the recently introduced comprehensive learning particle swarm optimizer (CLPSO) [63,64],

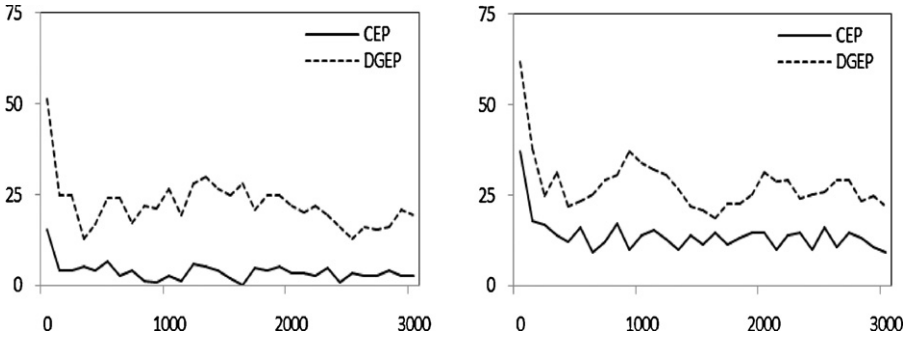

Fig. 7. Percentage of successful mutations by DGEP and CEP [2] on functions f_8 (left) and f_{12} (right). All results have been averaged over 50 independent runs.

Table 7
Comparison of successful mutation rates between DGEP and CEP [2] on the high dimensional classical benchmark functions. Results have been averaged over 50 independent runs. Best results are marked with boldface fonts.

Function	Successful mutation (%)		Function	Successful mutation (%)	
	DGEP	CEP		DGEP	CEP
f_1	**33.72**	25.08	f_8	**20.35**	5.19
f_2	**38.95**	15.05	f_9	**36.64**	14.64
f_3	**31.12**	22.46	f_{10}	**29.31**	22.94
f_4	**18.94**	10.21	f_{11}	**41.37**	32.18
f_5	**22.46**	11.25	f_{12}	**29.19**	14.59
f_6	**18.28**	7.48	f_{13}	**28.05**	16.36
f_7	**26.65**	17.19			

a variant of the particle swarm optimizer (PSO) [67]. CLPSO [63] uses a novel learning strategy in which all other particles' historical best information is used to update a particle's velocity to move the search process forward. It has demonstrated better performance than other variants of PSO for wide range of complex functions. Since the number of FEs used by CLPSO [63] is 200,000, DGEP is re-implemented for the same FEs. Table 8 presents results for DGEP and CLPSO [63] on six functions over 50 independent runs. DGEP performs better than CLPSO [63] on four functions, while CLPSO [63] outperforms DGEP on the remaining two only. The performance differences are clearly significant for all the functions.

4.2. DGEP on CEC2005 benchmark functions

DGEP is also applied to the new set of benchmark functions introduced in the special session on real-parameter optimization at CEC2005 [68]. The new set includes 25 functions with different complexity. These functions are challenging in the sense that many of them are the shifted, rotated, expanded or combined variants of classical benchmark functions. The last 11 functions of this benchmark set are the hybrid composite functions (CF_1–CF_{11}) that are the most challenging ones. In each of these composite functions CF_1–CF_{11}, a number of standard benchmark functions with randomly shifted global optimum and several randomly shifted strong local optima are combined in such a way that different functions' structures are blurred together. The performance of any evolutionary system is compromised more or less for all these functions. A more detailed description of these functions can be found in [68]. The dimension of all the functions is set to 30 and the FEs are set to be 3.0e+05. This setup is for a fair comparison with other works. The mean error values of 50 independent runs for DGEP, LSRCMA [26], NSDE [27,43] and CMAES [38] are presented in Table 9. Results indicate that DGEP achieves overall best performance compared to the other three algorithms on these functions. Out of these 11 functions, DGEP performs better than LSRCMA [26] on eight functions, shows similar performance on two while LSRCMA [26] performs better on only one function. In comparison with NSDE [27,28] on these 11 functions, DGEP performs better on six functions, shows similar performance on two while NSDE [27,28] performs better on the remaining three only. Also, DGEP outperforms CMAES [38] on nine out of these 11 functions while CMAES [38] performed better only on the remaining two. Out of these 11 functions, DGEP performs best on seven functions. Also, DGEP performs better or at least as well as any other algorithm on as many as nine functions out of these 11 functions. Thus the performance of DGEP is overall better than all of LSRCMA [26], NSDE [27,28] and CMAES [38].

5. Discussion

This section briefly explains why the performance of DGEP is often better than other existing algorithms in the comparison. Firstly, DGEP emphasizes both global explorations and local exploitations using the population diversity information. The utilization of the distance of dissimilar or similar chromosomes in mutation clearly reflects such emphasis. CEP [2], IFEP [6] and ALEP [5] do not separate exploration and exploitation operations; rather, IFEP [6] and ALEP [5] primarily emphasize producing good offspring. The emphasis on only good solutions may reduce the population diversity resulting in poor overall performance. IMGA [17,18] and RTS [52] alter the selection mechanism in such a way that it promotes diversity and allows better explorations, but at the cost of reduced exploitations and slower convergence speed. RCMA with XHC [25] performs exploration and exploitation separately by using PBX crossover [25] with BGA mutation [65] for exploration and a specialized crossover, XHC [25] for exploitation. The problem of using different operators lies in ensuring their synergistic effect [49]. DPGA [44] adapts the distance between its two populations in such a way that the reserve population mostly exerts exploitative pressure on the main population until the solutions get trapped into local optima for several generations. A more

Table 8
Comparison between DGEP and CLPSO [63] on six benchmark functions. Results have been averaged over 50 independent runs. Best results are marked with boldface fonts.

Function	DGEP	CLPSO	Function	DGEP	CLPSO
f_1	5.42×0^{-8}	$\mathbf{4.4 \times 10^{-14}}$	f_9	$\mathbf{1.56 \times 10^{-12}}$	4.8×10^{-10}
f_5	**1.28**	$2.1 \times 10^{+1}$	f_{10}	1.99×10^{-12}	**0.0**
f_8	$\mathbf{4.29 \times 10^{-12}}$	4.3×10^{-10}	f_{11}	$\mathbf{7.52 \times 10^{-14}}$	3.1×10^{-10}

Table 9
Comparison between DGEP, LSRCMA [26], NSDE [27,28] and CMAES [38] on 11 hybrid composition functions introduced at CEC2005. Results have been averaged over 50 independent runs. Best results are marked with boldface fonts.

Function	DGEP Mean error	LSRCMA Mean error	NSDE Mean error	CMAES Mean error
CF_1	4.88e+02	3.56e+02	3.64e+02	**2.16e+02**
CF_2	**3.72e+01**	3.26e+02	6.90e+01	5.84e+01
CF_3	1.62e+02	2.79e+02	**1.01e+02**	1.07e+03
CF_4	**5.49e+02**	8.77e+02	9.04e+02	8.90e+02
CF_5	**8.25e+02**	8.80e+02	9.04e+02	9.03e+02
CF_6	**7.91e+02**	8.79e+02	9.04e+02	8.89e+02
CF_7	5.00e+02	5.00e+02	5.00e+02	**4.85e+02**
CF_8	**8.41e+02**	9.08e+02	8.89e+02	8.71e+02
CF_9	**5.22e+02**	5.59e+02	5.34e+02	5.35e+02
CF_{10}	**2.00e+02**	**2.00e+02**	**2.00e+02**	1.41e+03
CF_{11}	2.05e+02	2.11e+02	**2.00e+02**	6.91e+02

balanced approach between exploitation and exploration might further improve its results. CLPSO [63] is a learning approach that does not employ exploration and exploitation operations separately. Although it utilizes the best information of all particles to update the velocity of any one particle, it may still trap into local optima due to many inherent problems of a learning scheme, e.g., difficulty in picking appropriate weight values to make a weighted combination of the old experiences and new observations. CMAES [38] maintains and continuously adapts a covariance matrix in order to maximize the likelihood of producing better offspring which makes the algorithm more exploitative rather than explorative in nature. The remaining two algorithms, NSDE [27,28] and LSRCMA [26] make attempts to combine the benefits of neighborhood search and local search techniques with differential evolution and RCMA respectively. However, the depth of local or neighborhood search on a chromosome is decided based on the fitness of the chromosome, which makes both these algorithms more exploitative than explorative. Lack of balance between exploitations and explorations tends to decrease population diversity and leads to premature convergence.

Secondly, mutation in DGEP does not produce offspring blindly, but rather utilizes the distance to other chromosomes in order to produce an offspring using an appropriate step size. This mutation produces an offspring in such a way that it either facilitates the exploration of wider regions of the search space or performs the exploitation of existing solutions for immediate fitness improvement. The mutation in CEP [2], IFEP [6], ALEP [5], IMGA [17,18] and RTS [52] does not use any information from other chromosomes and produces offspring blindly. The consequence of blind mutation is that the offspring produced may be dominated by chromosomes in the current population. RCMA with XHC [25] also uses blind mutation and crossover for exploration. All of DPGA [44], NSDE [27,28], LSRCMA [26] and CMAES [38] make use of population information either to guide the depth of genetic operations on each chromosome or to adapt step size values for increasing the likelihood of successful mutations which help them achieve better results than CEP [2], IFEP [6] and ALEP [5] and sometimes competitive results to DGEP.

Thirdly, DGEP uses selection strategies at two different levels in order to select offspring for the next generation. During each mutation, it selects an offspring to choose between exploration and exploitation. After mutating all the chromosomes, it performs a fitness based selection to pick the best chromosomes for the next generation. This two-level selection strategy is more likely to perform a better balance between exploitative and explorative features of the evolutionary search than the more exploitative fitness based schemes of IMGA [17,18], RTS [52], DPGA [44], NSDE [27,28], LSRCMA [26] and CMAES [38]. Also, the diversity preservation schemes of DGEP ensure adequate amount of population diversity throughout the evolution to assist its diversity guided mutation scheme. In case of CEP [2], ALEP [5] and IFEP [6], a tournament based selection scheme is adopted where for each parent or offspring, q opponents are chosen uniformly at random for pair wise fitness comparison from all the parents and offspring. The value of q affects the population diversity. A large value of q corresponds to high selection pressure, so the probability of the fittest chromosome being selected multiple times becomes high, resulting in loss of population diversity. RCMA with XHC [25] allows only better offspring for both exploration and exploitation, thus it tends to reduce population diversity, which is the main reason for premature convergence.

The incorporation of few simple ideas may further improve the results of DGEP, especially on the complex rotated and hybrid composite functions. First, a simple random value of K is used for the diversity guided mutation. An adaptive approach that can dynamically change K for explorations or exploitations during the course of evolution may be more appropriate than a simple random strategy. Second, DGEP showed remarkable explorative performance to locate the global optima, but its exploitative capacity is not as good as some other existing approaches, such as DPGA [44]. It would be interesting to observe if DGEP could be hybridized with other algorithms, e.g., DPGA [44] or similar algorithms having better fine tuning capacity. Third, the mutation step size is guided by considering the distance information of only two neighboring or distant chromosomes. It could be better if the distance information involving more chromosomes, e.g., the relative density of chromosomes across the search space or the characteristics of the fitness landscape around the current point could be considered to extract better guidance for the mutation step size. Fourth, DGEP has been applied on continuous optimization problems only. It would be interesting to study how well DGEP performs for other problems, especially the real world ones. The incorporation of these ideas could be a topic for our future study.

References

[1] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
[2] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, New York, 1995.
[3] H.-P. Schwefel, Numerical Optimization of Computer Models, John Wiley & Sons, Chichester, UK, 1981.
[4] J. Yao, N. Kharma, P. Grogono, Bi-objective multipopulation genetic algorithm for multimodal function optimization, IEEE Trans. Evol. Comput. 14 (1) (2010) 80–102.
[5] C. Lee, X. Yao, Evolutionary programming using mutations based on the Lévy probability distribution, IEEE Trans. Evol. Comput. 8 (2004) 1–13.
[6] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[7] B. Soylu, M. Koksalan, A favorable weight-based evolutionary algorithm for multiple criteria problems, IEEE Trans. Evol. Comput. 14 (2) (2010) 191–205.
[8] E. Zitzler, L. Thiele, J. Bader, On set-based multiobjective optimization, IEEE Trans. Evol. Comput. 14 (1) (2010) 58–79.
[9] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, Multiobjective genetic algorithm-based fuzzy clustering of categorical attributes, IEEE Trans. Evol. Comput. 13 (5) (2009) 991–1005.
[10] W.-N. Chen, J. Zhang, H.S.H. Chung, W.-L. Zhong, W.-G. Wu, Y.-h. Shi, A novel set-based particle swarm optimization method for discrete optimization problems, IEEE Trans. Evol. Comput. 14 (2) (2010) 278–300.
[11] P.S. Oliveto, J. He, X. Yao, Analysis of the (1+1)-EA for finding approximate solutions to vertex cover problems, IEEE Trans. Evol. Comput. 13 (5) (2009) 1006–1029.
[12] J.S. Aguilar-Ruiz, J.C. Riquelme, M. Toro, Evolutionary learning of hierarchical decision rules, IEEE Trans. Syst. Man Cybern. B: Cybern. 33 (2003) 324–331.
[13] G. Schneider, H. Wersing, B. Sendhoff, Edgar Körner, Evolutionary optimization of a hierarchical object recognition model, IEEE Trans. Syst. Man Cybern. B: Cybern. 35 (2005) 426–437.
[14] K.M. Sim, Evolving fuzzy rules for relaxed-criteria negotiation, IEEE Trans. Syst. Man Cybern. B: Cybern. 38 (2008) 1486–1500.
[15] O. Buchtala, M. Klimek, B. Sick, Evolutionary optimization of radial basis function classifiers for data mining applications, IEEE Trans. Syst. Man Cybern. B: Cybern. 35 (2005) 928–947.

[16] S.-J. Han, S.-B. Cho, Evolutionary neural networks for anomaly detection based on the behavior of a program, IEEE Trans. Syst. Man Cybern. B: Cybern. 36 (2006) 559–570.
[17] T. Bäck, D.B. Fogel, Z. Michalewicz, et al. (Eds.), Handbook on Evolutionary Computation, IOP Publishing Ltd./Oxford University Press, 1997.
[18] T. Jumonji, G. Chakraborty, H. Mabuchi, M. Matsuhara, A novel distributed genetic algorithm implementation with variable number of islands, in: Proc. IEEE Congr. Evolut. Comput., 2007, pp. 4698–4705.
[19] R.K. Ursem, Multinational evolutionary algorithms, in: P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, A. Zalzala (Eds.), Proc. Congress of Evolutionary Computation (CEC-99), vol. 3, 1999, pp. 1633–1640.
[20] R. Thomsen, P. Rickers, T. Krink, Religion-based spatial model for evolutionary algorithms, in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN VI, vol. 1, 2000, pp. 817–826.
[21] B. Sareni, L. Krahenbuhl, Fitness sharing and niching methods revisited, IEEE Trans. Evol. Comput. 2 (3) (1998 Sep) 97–106.
[22] K. Deb, D. Goldberg, An investigation of niche and species formation in genetic function optimization, in: Proc. 3rd Int. Conf. Genetic Algorithm (ICGA), 1989, pp. 42–50.
[23] A. Della Cioppa, C. DeStefano, A. Marcelli, On the role of population size and niche radius in fitness sharing, IEEE Trans. Evol. Comput. 8 (6) (2004) 580–592.
[24] A. Della Cioppa, C. DeStefano, A. Marcelli, Where are the niches? Dynamic fitness sharing, IEEE Trans. Evol. Comput. 11 (4) (2007) 453–465.
[25] M. Lozano, F. Herrera, N. Krasnogor, D. Molina, Real-coded memetic algorithms with crossover hill-climbing, Evol. Comput. 12 (2004) 273–302.
[26] D. Molina, F. Herrera, M. Lozano, Adaptive local search parameters for real-coded memetic algorithms, in: Proc. IEEE Congress Evolutionary Computation (CEC 2005), 2005, pp. 888–895.
[27] Z. Yang, J. He, X. Yao, Making a difference to differential evolution, in: Z. Michalewicz, P. Siarry (Eds.), Advances in Metaheuristics for Hard Optimization, Springer, 2007, pp. 397–414.
[28] Z. Yang, K. Tang, X. Yao, Differential evolution for high-dimensional function optimization, in: Proc. 2007 IEEE Congress on Evolutionary Computation (CEC'07), Singapore, 25–28 September, 2007, pp. 3523–3530.
[29] M. Kominami, T. Hamagami, A new genetic algorithm with diploid chromosomes by using probability decoding for nonstationary function optimization, in: Proc. IEEE Int. Conf. Syst., Man, Cybern., 2007, pp. 1268–1273.
[30] Y. Yoshida, N. Adachi, A diploid genetic algorithm for preserving population diversity-pseudo-Meiosis GA, in: Proc. 3rd Parallel Problem Solving Nature (PPSN), 1994, pp. 36–45.
[31] I. Rechenberg, Evolutions Strategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog Verlag, Stuttgart, 1973.
[32] H.P. Schwefel, Numerical Optimization of Computer Models, John Wiley & Sons, Chichester, UK, 1981.
[33] J.D. Schaffer, R. Caruana, L.J. Eshelman, R. Das, A study of control parameters affecting on-line performance of genetic algorithms for function optimisation, in: Proc. 3rd Int. Conf. Genetic Algorithms, 1989, pp. 51–60.
[34] W. Spears, Crossover or Mutation in Foundations of Genetic Algorithms, vol. II, Morgan Kaufmann, Whitley, 1992.
[35] T. Bäck, The interaction of mutation rate, selection and self-adaptation within a genetic algorithm, in: Manderick Manner (Ed.), Parallel Problem Solving from Nature, Elsevier Science, 1992, pp. 85–94.
[36] F. Hoffmeister, T. Bäck, Genetic self learning, in: Varela, Bourgine (Eds.), Towards a Practice on Autonomous Systems, MIT Press, 1992, pp. 227–235.
[37] T. Bäck, Self adaptation in genetic algorithms, in: Varela, Bourgine (Eds.), Towards a Practice on Autonomous Systems, MIT Press, 1992, pp. 263–271.
[38] N. Hansen, S.D. Muller, P. Koumoutsakos, Reducing the time complexity of the de-randomized evolution strategy with covariance matrix adaptation (CMA-ES), Evol. Comput. 11 (2003) 1–18.
[39] T.C. Fogarty, Varying the probability of mutation in the genetic algorithm, in: Proc. 3rd International Conference on Genetic Algorithms, Morgan Kaufmann, Schaffer, 1989, pp. 104–109.
[40] J. Hesser, R. Männer, Investigation of the m-heuristic for optimal mutation probabilities, in: Proc. 2nd Parallel Problem Solving Nature (PPSN), 1992, pp. 115–124.
[41] D. Thierens, Adaptive mutation rate control schemes in genetic algorithms, in: Proc. Congr. Evol. Comput., vol. 1, 2002, pp. 980–985.
[42] R.K. Ursem, Diversity guided evolutionary algorithm, in: J.J. Merelo, P. Adamidis, H.-P. Schwefel (Eds.), Proc. Parallel Problem Solving from Nature (PPSN) VII, vol. 2439, Granada, Spain, 2002, pp. 462–471.
[43] S. Tsutsui, Y. Fujimoto, Forking genetic algorithm with blocking and shrinking modes (FGA), in: Proc. Int. Conf. Genetic Algorithms, 1993, pp. 206–215.
[44] T. Park, K.R. Ryu, A dual-population genetic algorithm for adaptive diversity control, IEEE Trans. Evol. Comput. 14 (6) (2010) 865–884.
[45] E. Cantúz-Paz, Migration policies, selection pressure, and parallel evolutionary algorithms, J. Heuristics 7 (4) (2001) 311–334.
[46] G. Rudolph, On takeover times in spatially structured populations: array and ring, in: Proc. 2nd Asia-Pacific Conf. Genetic Algorithms Applicat., 2000, pp. 144–151.
[47] I. Sekaj, Robust parallel genetic algorithms with re-initialization, in: Proc. 8th Parallel Problem Solving Nature (PPSN), 2004, pp. 411–419.
[48] Z. Skolicki, K. De Jong, The influence of migration sizes and intervals on island models, in: Proc. Genetic Evol. Comput. Conf. (GECCO), 2005, pp. 1295–1302.
[49] S. Mahfoud, Crowding and preselection revisited, Illinois Genetic Algorithms Laboratory (IlliGAL). Technical Report 92004, 1992.
[50] S.W. Mahfoud, Niching methods for genetic algorithms, Ph.D. dissertation, Dept. General Eng., Univ. Illinois, Urbana-Champaign, 1995.
[51] O.J. Mengsheol, D.E. Goldberg, Probabilistic crowding: deterministic crowding with probabilistic replacement, in: Proc. Genetic Evol. Comput. Conf. (GECCO), 1999, pp. 409–416.
[52] G. Harick, Finding multimodal solutions using restricted tournament selection, in: Proc. 6th Int. Conf. Genetic Algorithms (ICGA), 1995, pp. 24–31.
[53] H. Shimodaira, A diversity control oriented genetic algorithm (DCGA): development and experimental results, in: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (Eds.), Proc. Genetic Evol. Comput. Conf., 1999, pp. 603–611.
[54] T. Park, K.R. Ryu, Crew pairing optimization by a genetic algorithm with unexpressed genes, J. Intell. Manuf. 17 (4) (2006) 375–383.
[55] P. Collard, J.P. Aurand, DGA: an efficient genetic algorithm, in: Proc. 11th Eur. Conf. Artif. Intell., 1994, pp. 487–491.
[56] P. Collard, C. Escazut, Genetic operators in a dual genetic algorithm, in: Proc. 7th Int. Conf. Tools Artif. Intell., 1995, pp. 12–19.
[57] S. Yang, PDGA: The Primal–Dual Genetic Algorithm in Design and Application of Hybrid intelligent Systems, Ios Press, Amsterdam, The Netherlands, 2003, pp. 214–223.
[58] S. Yang, Nonstationary problem optimization using the primal–dual genetic algorithm, in: Proc. Congr. Evol. Comput., vol. 3, 2003, pp. 2246–2253.
[59] T. Bäck, H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, Evol. Comput. 1 (1) (1993) 1–23.
[60] K.A. De Jong, An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1975. Dissertation Abstracts International 36(10), 5140B, University Microfilms Number 76-9381.
[61] D.E. Goldberg, J. Richardson, Genetic algorithms with sharing for multimodal function optimization, in: J.J. Grefenstette (Ed.), Genetic Algorithms and their Applications – ICGA'87, Lawrence Erlbaum Associates Publishers, 1987, pp. 41–49.
[62] H.G. Cobb, J.F. Grefenstette, Genetic algorithms for tracking changing environments, in: S. Forrest (Ed.), Proc. 5th International Conference on Genetic Algorithms, 1993, pp. 523–530.
[63] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006) 281–295.
[64] J.J. Liang, P.N. Suganthan, K. Deb, Novel composition test functions for numerical global optimization, in: Proc. IEEE Swarm Intelligence Symposium, 2005, pp. 68–75.
[65] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm for continuous parameter optimization, Evol. Comput. 1 (1993) 25–49.
[66] D.K. Gehlhaar, D.B. Fogel, Tuning evolutionary programming for conformationally flexible molecular docking, in: L.J. Fogel, P.J. Angeline, T. Back (Eds.), Evolutionary Programming V: Proc. Fifth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1996, pp. 419–429.
[67] C.A. Coello Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput. 8 (2004) 256–279.
[68] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, in: Nanyang Technological University, Singapore, Technical Report, 2005 and IIT Kanpur, India, KanGAL Report No. 2005005, 2005.
[69] K. Deb, A. Anand, D. Joshi, A computationally efficient evolutionary algorithm for real-parameter optimization Evol. Comput. 10 (4), 371–395.