Applied Mathematics and Computation 283 (2016) 181–194

Contents lists available at ScienceDirect

Applied Mathematics and Computation

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

Epistatic arithmetic crossover based on Cartesian graph product in ensemble differential evolution

Iztok Fister[a], Aleksandra Tepeh[a,b], Iztok Fister Jr.[a,*]

[a] Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia
[b] Faculty of Information Studies, Ljubljanska cesta 31A, SI-8000 Novo Mesto, Slovenia

ARTICLE INFO

Keywords:
Cartesian graph product
Optimization
Epistatic arithmetic crossover
Ensemble strategies
Differential evolution

ABSTRACT

Epistasis in genetics denotes an impact of one gene on the expression of the another genes. This means that so called epistatic gene influences the characteristics of the so called hypostatic genes. As a matter of fact, there is no one-to-one correspondence between genes and traits in nature. On the other hand, values of offspring genes are inherited from the parents genes. In this paper, the impact of epistatic genes in evolutionary computation is studied, where each epistatic gene in offspring depends on the corresponding hypostatic genes of its parents by an epistatic arithmetic crossover used in differential evolution. Thus, epistatic genes are determined by the Cartesian graph product of both parents presented as linear graphs. The epistatic arithmetic crossover is applied as a mutation strategy to the ensemble differential evolution. The results of extensive experiments conducted on CEC-14 function benchmark suite showed a great potential of the proposed algorithm and encouraged us to start to experiment with other graph products as well.

1. Introduction

Evolutionary algorithms (EAs) represents nowadays a powerful tool for solving the hardest problems with which humans are confronted today. They found an inspiration for their operation in Darwinian evolution [15], where the fittest individuals in a population have more chances for surviving and transferring the best characteristics on their offspring. Modern biochemistry and genetics have extended this Darwinian macroscopic view on evolution by microscopic view concerning the mechanisms of heredity [16].

In genetics, a genotype determines a characteristics (traits) of each individuals (i.e., its phenotype) and represents a blueprint or set of instructions for building and maintaining a living cell. Thus, all information are written in genes that represent transfer units of heredity. A sequence of genes is combined into a linear order known as chromosome. Each gene is positioned at the specific chromosomal position (also loci). Interestingly, a mapping of genes to traits is not one-to-one (not injective). Although one gene can determine one traits, in common, more genes in chromosome specify one trait. In line with this, the fitness of individual trait depends on contribution that one gene has on the value of another gene. This dependence is also known under the name epistasis, where the more epistatic genes influence the expression of one hypostatic gene.

* Corresponding author. Tel.: +386 31419573.
E-mail addresses: iztok.fister@um.si (I. Fister), aleksandra.tepeh@um.si (A. Tepeh), iztok.fister1@um.si (I. Fister Jr.).

http://dx.doi.org/10.1016/j.amc.2016.02.034
0096-3003/

The phenomenon of epistasis in the genetic algorithms was the first considered by Davidor in [3,4] that connected each problem with its so called epistatic level. This level corresponds to the maximal number of dependent variables k of n representing a solution of some sub-problems into which the problem is decomposed. For instance, the problem with epistatic level zero cannot be decomposed into sub-problems, because all n variables in the solution are independent. In general, these k variables represent a block that can be overlapped with another block by sharing some variables. The classical problem of size n and epistatic level k with overlapping blocks is the *NK*-landscape [6]. The more complete survey of this domain can be found in Gras [5].

Mathematically, a set of epistatic genes can be determined in different ways. In our study, we use a tool from graph theory, more specifically, the Cartesian product of graphs. It turns out that many real-world problems can conveniently be described by means of a graph consisting of a set of nodes (vertices) together with lines (edges) joining certain pairs of these nodes [38–40,51]. Graphs represent invaluable models for better understanding of biological systems. Nodes in biological networks can represent biomolecules such as genes, proteins or metabolites, and edges connecting these nodes indicate functional, physical or chemical interactions between the corresponding biomolecules. The Cartesian product is a binary operation which creates a new graph from two initial ones, and in many cases properties of a Cartesian product graph can be derived from the properties of its factors. Cartesian products of graphs have applications in many branches, like coding theory, network designs, chemical graph theory, telecommunications, and others [41–43,50]. In this study the Cartesian product is used to determine epistatic genes, where the factors of this product are paths (linear graphs) representing both parents.

Differential evolution (DE) developed by Storn and Price in 1995 [7] has become one of the most prominent EAs for solving real-world optimization problems. Like other standard EAs, i.e., genetic algorithms (GA) [17], genetic programming (GP) [18], Evolution Strategies (ES) [16] and Evolutionary Programming (EP) [19], the DE is also population-based algorithm, where each individual in the population represents a solution of the problem to be solved in a form of real-vectors. These vectors (also parents) are subject of operators crossover and mutation. As a result, trial vectors (also offspring) are produced that compete with their parents for a place in the next generation. In this competition for surviving, the better among parent and a trial vector according to the fitness value is placed at the parent's position in the population.

In order to improve the algorithm, a lot of modifications have been applied to the original DE. Mainly, modifications concerned setting DE algorithm parameters and changing DE mutation strategies. Algorithm parameters control the evolutionary search process, because the parameters valid at the beginning become inappropriate at the end of the optimization and vice versa. Therefore, the DE parameters need to be adapted and self-adapted. The more successful algorithms of this kind are jDE [8] and SaDE [10]. The DE mutation strategies determine a way in which the search space is explored. As it is known, some DE mutations are more explorative, while the others are more exploitative. As a result, the mutation strategies need to be changed during a discovering of the search space. This kind of adaptation has previously been addressed in [11] and widened with so called arithmetic recombination as a mutation strategy in ensemble DE strategies in [20] based on linear arithmetic crossover as developed for GAs in [21]. Finally, some kind of DE algorithms try to improve the results of the original DE algorithm by using ensembles of parameters and mutation DE strategies [12,13]. A complete survey of DE methods can be found in [1,14].

In this study, the epistatic differential crossover based on Cartesian graph product in ensemble DE (eXEDE) is proposed, where the epistatic arithmetic crossover is used as a strategy in ensemble DE strategies that beside the ordinary arithmetic crossover [2] takes into account also interactions with epistatic genes. While the original linear crossover calculates three possible values for two parameters laying at the same positions in parent vectors, the epistatic arithmetic crossover selects one of these three values randomly and to this value adds contributions of the epistatic genes determined by Cartesian graph products. As a result, one offspring is generated. Let us notice that these contributions consist of two parts, i.e., dependence contribution of epistatic genes of the first as well as the dependence contribution of the second parent. However, the magnitude of both contributions that have an impact on the final value of a specific parameter can be regulated by weights. The corresponding eXEDE was applied to the CEC-14 function benchmark suite. The results of the optimization showed that the proposed eXEDE improved the results of the original DE, both self-adaptive DEs (i.e., jDE and SaDE), ensemble DE (EDE) as well as the ensemble DE with arithmetic crossover (XEDE).

The structure of the remainder of the paper is as follows. Section 2 discusses background information about foundations of Cartesian graph products and differential evolution. In Section 3, an evolution of arithmetic crossover strategies in differential evolution is given. The experiments and results are subjects of Section 4. Section 5 concludes the paper with summarizing of the performed work and the possible directions for further development are outlined.

2. Background information

2.1. Cartesian graph products

The *Cartesian product* $G\Box H$ of graphs G and H is the graph with the vertex set $V(G) \times V(H)$ in which vertices (g, h) and (g', h') are adjacent whenever $gg' \in E(G)$ and $h = h'$, or $g = g'$ and $hh' \in E(H)$. Thus, $V(G)$ and $V(H)$ denote the vertex sets of graphs G and H, respectively, while the $E(G)$ and $E(H)$ their corresponding edge sets.

According to Imrich and Klavžar [25], Cartesian products of graphs were defined in 1912 by Whitehead and Russell. Later they were repeatedly rediscovered and studied extensively since the 1960s. Due to their nice metric properties, they have

found numerous applications in coding theory, radio-frequency assignment, theoretical chemistry, etc. The most important metric property of the Cartesian product operation is that for any graphs G and H,

$$d_{G\Box H}((g,h),(g',h')) = d_G(g,g') + d_H(h,h').$$

It follows immediately from the definition that the Cartesian product is commutative and associative, i.e., $G\Box H = H\Box G$ and $G\Box(H\Box K) = (G\Box H)\Box K$ (up to isomorphism) for all graphs G, H and K, with K_1 as its unit. The product $G\Box H$ is connected if and only if both factors G and H are connected.

A graph G is *prime* if it is nontrivial and $G = G_1\Box G_2$ implies $G_1 = K_1$ or $G_2 = K_1$. Every graph G has a prime factorization $G = G_1\Box G_2\Box\ldots\Box G_p$, where each factor G_i is prime. A fundamental theorem, proved independently by Sabidussi [27] and Vizing [28], states that the prime factorization of a connected graph is unique, that is, if a connected graph G has prime factorizations $G_1\Box G_2\Box\ldots\Box G_p$ and $H_1\Box H_2\Box\ldots\Box H_q$, then $p = q$ and $G_i = H_i$ for $1 \le i \le p$ (after reindexing, if necessary).

This fundamental theorem was a starting point for research concerning the relations between a Cartesian product and its factors. These relations are of particular interest as they allow us to break down problems by transferring algorithmic complexity from the product to the factors. In 2006, Imrich and Peterin [26] gave an algorithm able to compute the prime factorization of connected graphs in linear time and space, making the use of Cartesian product decomposition particularly attractive.

Prime examples of Cartesian product graphs are hypercubes. A hypercube, i.e., a Cartesian product of two-vertex complete graphs, is one of the most widely used topologies because it provides small diameter and embedding of various interconnection networks. In [32] the authors show that simultaneous broadcasting in hypercube networks via mutually independent Hamiltonian cycles is robust with respect to edge faults. In [34] the authors significantly improve previously known results on queue layouts of hypercubes, which is an interesting concept with applications in sorting permutations, parallel process scheduling, matrix computations, graph drawings, and queue-based computers. Gray codes [37], which can be viewed as Hamiltonian paths of hypercubes, are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems. Research on Gray codes satisfying certain additional properties [35,36] has received a considerable attention, and applications have been found in diverse areas like data compression, graphics and image processing, information retrieval, signal encoding, processor allocation in hypercubic networks. A general construction of level-disjoint partitions for Cartesian products which allows simultaneous broadcasting of optimal number of messages in optimal time in some of these networks was developed in [33]. Hypercubes and other general Cartesian products of graphs have found applications in many other different fields of science.

In this paper we use 2-dimensional grids, i.e., Cartesian products of two paths, where a path (or linear graph) on n vertices P_n is given by the set of vertices $V(P_n) = \{v_1, v_2, \ldots, v_n\}$ and the set of edges $E(P_n) = \{v_i v_{i+1} | i = 1, \ldots, n-1\}$.

The *open neighborhood* of a vertex g in G is defined as $N_G(g) = \{g'; gg' \in E(G)\}$. This means that for the product $G\Box H$ and a vertex $(g,h) \in V(G\Box H)$ we have: $N_{G\Box H}(g,h) = \{(x,y); (x,y)(g,h) \in E(G\Box H)\}$. It follows from the definition that the open neighborhood of a vertex $(g,\ h)$ in the Cartesian product $G\Box H$ equals $N_{G\Box H}(g,h) = \{(x,h); x \in N_G(g)\} \cup \{(g,y); y \in N_H(h)\}$. Vertices from open neighborhoods are used in the function, defined in Section 3.

2.2. Differential evolution

Differential evolution (DE) [1] is an evolutionary algorithm appropriate for continuous and combinatorial optimization that was introduced by Storn and Price in 1995 [7]. This is a population-based algorithm that consists of Np vectors with real-coded parameters of dimensions D representing the candidate solutions, as follows:

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, \ldots, x_{i,D}^{(t)}), \quad \text{for} \quad i = 1, \ldots, Np, \tag{1}$$

where t denotes the generation number.

The variation operator in DE supports a differential mutation and a differential crossover. In particular, the differential mutation randomly selects two solutions and adds a scaled difference between these to the third solution. This mutation can be expressed as follows:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r1}^{(t)} + F \cdot (\mathbf{x}_{r2}^{(t)} - \mathbf{x}_{r3}^{(t)}), \quad \text{for} \quad i = 1, \ldots, Np, \tag{2}$$

where F denotes the scaling factor as a positive real number that scales the rate of modification whilst $r1, r2, r3$ are randomly selected values in the interval $1, \ldots, Np$. Note that Price and Storn proposed $F \in [0.0, 2.0]$ in the original DE, but typically the interval $F \in [0.1, 1.0]$ is used in the DE community.

Uniform crossover is typically used in DE, where the trial vector is built from parameter values copied from two different solutions. Mathematically, this crossover can be expressed as follows:

$$w_{i,j}^{(t+1)} = \begin{cases} u_{i,j}^{(t)} & \text{rand}_j(0,1) \le CR \vee j = j_{rand}, \\ x_{i,j}^{(t)} & \text{otherwise}, \end{cases} \tag{3}$$

where $CR \in [0.0, 1.0]$ controls the fraction of parameters that are copied to the trial solution. Note that the relation $j = j_{rand}$ ensures that the trial vector is different from the original solution $\mathbf{x}_i^{(t)}$.

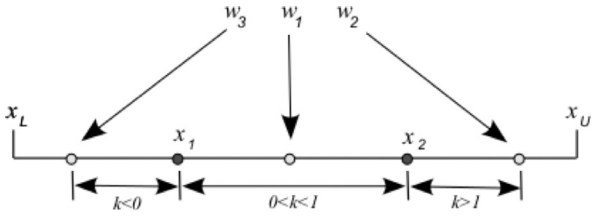


Fig. 1. The generated mutation element w_1 of vector $\mathbf{w}_1$ lies between elements x_1 and x_2 of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, when $0 < K < 1$, element w_2 of vector $\mathbf{w}_2$ above the element x_2 of vector $\mathbf{x}_2$ when $K > 1$ and element w_3 of vector $\mathbf{w}_3$ below the element w_3 of vector $\mathbf{w}_3$ for $K < 0$.

A differential selection is in fact a generalized one-to-one selection that can be mathematically expressed as follows:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)} & \text{if } f(\mathbf{w}_i^{(t)}) \leq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)} & \text{otherwise.} \end{cases} \tag{4}$$

In a technical sense, crossover and mutation can be performed in several ways in differential evolution. Therefore, a specific notation is used to describe the varieties of these methods (also strategies) generally. For example, 'rand/1/bin' denotes that the base vector is randomly selected, 1 vector difference is added to it, and the number of modified parameters in the mutant vector follows binomial distribution. A detailed description of the other DE mutation strategies as well as exponential crossover can be seen in [1,22].

3. Evolution of arithmetic crossover strategies

At first glance, using the arithmetic crossover as a mutation strategy in DE sounds very confusing. It is about a mixture of two different notations, i.e., the one used in EA community and the other in DE community. In EA sense, the crossover means a way in which parameters of one parent are exchanged with at the same position layed parameters of the second parent, while the crossover in DE prescribes how many parameters will be exchanged between both parameters. The method of changing the parameters is defined by mutation strategy in DE. Therefore, a concept arithmetic crossover in this paper denotes a method of changing parameters and indeed represents a mutation strategy with crossover rate $CR = 1.0$ (i.e., there is no crossover in DE sense).

3.1. Arithmetic crossover strategy

The arithmetic crossover strategy was developed by Hui and Suganthan [20] and it is based on linear crossover reported by Wright [21]. From two parent solutions $\mathbf{x}_i$ and $\mathbf{x}_j$, the linear crossover creates the three solutions, i.e.,

$$\begin{aligned} \mathbf{w}_1 &= 0.5(\mathbf{x}_i^{(t)} + \mathbf{x}_j^{(t)}), \\ \mathbf{w}_2 &= (1.5\mathbf{x}_i^{(t)} - 0.5\mathbf{x}_j^{(t)}), \\ \mathbf{w}_3 &= (-0.5\mathbf{x}_i^{(t)} + 1.5\mathbf{x}_j^{(t)}). \end{aligned} \tag{5}$$

at generation t. However, only two solutions are selected by this operator. Using some perturbations, the proposed Eq. (5) can be written as follows

$$\begin{aligned} \mathbf{w}_1 &= \mathbf{x}_i^{(t)} + 0.5(\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)}), \\ \mathbf{w}_1 &= \mathbf{x}_i^{(t)} - 0.5(\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)}), \\ \mathbf{w}_1 &= \mathbf{x}_i^{(t)} + 1.5(\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)}). \end{aligned} \tag{6}$$

When the factors 0.5 and 1.5 in Eq. (6) are replaced with K and indices i and j with random vectors $r1$ and $r2$, the following equation is obtained

$$0w_i = x_{r1} + K(x_{r2} - x_{r1}), \tag{7}$$

which represents the arithmetic crossover strategy. A graphical representation of this crossover for vectors in 1-dimensional space is presented in Fig. 1.

If only one solution needs to be selected from three different intervals, the value K must be selected from the interval $[K_L, K_U]$, where the $K_L < 0$ and $K_U > 1$. However, the setting of proper value of this parameter has a great influence on the performance of this arithmetic crossover strategy, because both parameters determine a magnitude of the search space to be explored.

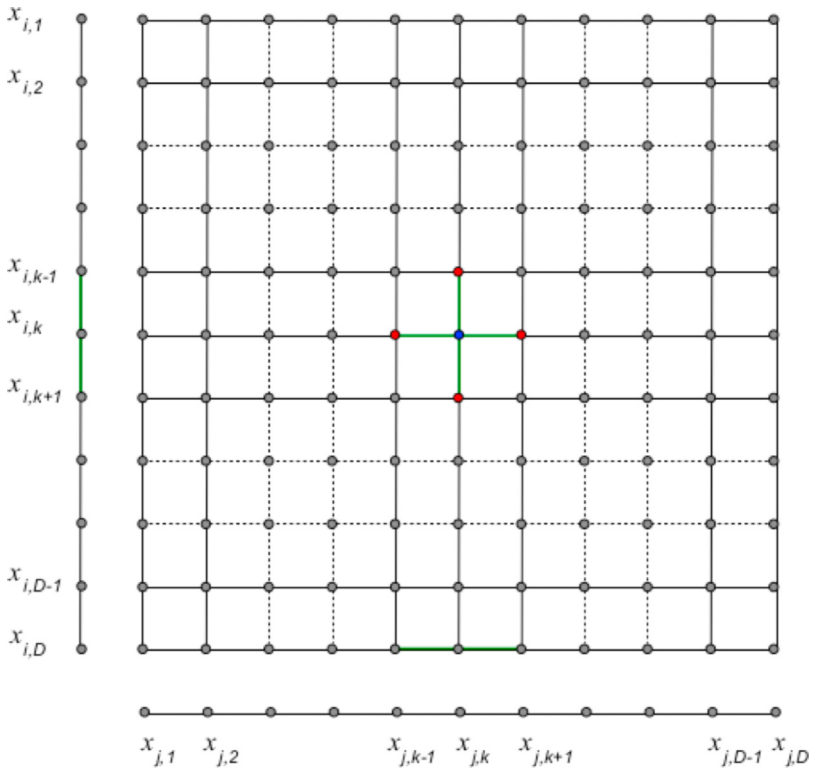


Fig. 2. The figure represents the neighborhood of the kth element obtained as the Cartesian graph product of linear graphs $G(\mathbf{x}_i)\square G(\mathbf{x}_j)$. This element is denoted in blue color, while the corresponding epistatic neighbors in red color. Edges that connect neighbors with the kth element are colored with green color. Thus, the epistatic elements $x_{i,k-1}$ and $x_{i,k+1}$ depend more on the linear graph $G(\mathbf{x}_i)$, while elements $x_{j,k-1}$ and $x_{j,k+1}$ more on the linear graph $G(\mathbf{x}_j)$. However, the influence of the epistatic elements are controlled by epistatic coefficient. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2. Epistatic arithmetic crossover strategy

Epistasis in genetics addresses genes having a contribution on the phenotype traits of individuals. This phenomenon in evolutionary computation community denotes the effect of a unit that is context dependent, i.e., one unit is not predictable unless the value of another unit is known [23]. Epistasis is related to the problem difficulty of EAs.

A basis of the epistatic arithmetic crossover represents an arithmetic crossover strategy presented in Eq. (1) to which a contribution of adjacent elements of vectors is determined by Cartesian graph product. Thus, both parents $\mathbf{x}_i$ and $\mathbf{x}_j$ entering in a process of a crossover are treated as linear graphs $G(\mathbf{x}_i)$ and $G(\mathbf{x}_j)$, where edges in the Cartesian graph product $G(\mathbf{x}_i)\square G(\mathbf{x}_j)$ determine the neighborhood for each vertex in this graph (Fig. 2).

The neighborhood defines the epistatic elements (genes) which contribute by forming the total value of the corresponding offspring element. Let us notice that the contribution consists of two parts, i.e., the contribution of the first parent $\mathbf{x}_i$ and the second parent $\mathbf{x}_j$ expressed as

$$\begin{aligned}\Delta_{i,k} &= \delta_{i,k-1}(x_{j,k} - x_{i,k-1}) + \delta_{i,k}(x_{j,k} - x_{i,k+1}),\\ \Delta_{j,k} &= \delta_{j,k-1}(x_{j,k-1} - x_{i,k}) + \delta_{j,k}(x_{j,k+1} - x_{i,k}),\end{aligned} \tag{8}$$

where $\delta_{i,\,k}$ determines the dependence of the kth epistatic element $x_{i,k+1}$ on the hypostatic element $x_{i,\,k}$ in vector $\mathbf{x}_i$ and $\delta_{j,\,k}$ denotes the dependence of the kth epistatic element $x_{j,k+1}$ on the same hypostatic element.

Dependence vectors $\boldsymbol{\delta}_i = \{\delta_{i,k}\}$ for $i = 1, \ldots, NP$ and $k = 1, \ldots, D$ are realized as a circular linked list, whose elements are self-adapted according to equation

$$\delta_{i,k}^{(t+1)} = \delta_{i,k}^{(t)} \exp^{\tau N(0,1)}, \tag{9}$$

where $\tau = 1/\sqrt{D}$ is a learning rate and $N(0,\,1)$ designates a random number drawn from normal distribution with mean zero and standard deviation one. One dependence vector $\boldsymbol{\delta}_i$ is added to each solution vector $\mathbf{x}_i$ by the epistatic arithmetic crossover. Self-adaptation can take place when so called evolution window is found, where the evolutionary search can progress [16]. The occurrence of this window is connected with a proper setting the starting value $\delta_{i,k}^{(0)}$ for $i = 1, \ldots, NP \wedge j = 1, \ldots, D$ that usually demands a lot of experimental work in order to be determined.

In Eq. (8), $\Delta_{i,k}$ and $\Delta_{j,k}$ denote a total dependence of epistatic elements by both parents $\mathbf{x}_i$ and $\mathbf{x}_j$ on the kth element of offspring according to the equation

$$\mathbf{w}_i = \mathbf{x}_{r1} + K(\mathbf{x}_{r2} - \mathbf{x}_{r1}) + E\boldsymbol{\Delta}_{r1} + (1-E)\boldsymbol{\Delta}_{r2}, \tag{10}$$

where the parameter $E \in [0, 1]$ is so-called epistatic coefficient that regulates an epistatic impact of both parents on their offspring.

Let us notice that the epistatic arithmetic crossover (Eq. (10)) is distinguished from the arithmetic crossover presented in Eq. (7) regarding the last two terms, where contributions of epistatic elements on the total values of hypostatic elements of a parent $\mathbf{x}_{r1}$ are considered additionally [24].

3.3. Ensemble DE using the epistatic arithmetic crossover strategy

Mutation DE strategies describe a kind of a problem search space exploration. Typically, this exploration is a dynamic process in which an evolutionary search is adapted to a fitness landscape. The fitness landscape is a metaphor borrowed from the genetics [29], where genotype characteristics represent a search space, while phenotype determines a fitness. Each point in a search space comply with the corresponding fitness point. All fitness points describe the fitness landscape, while a specific fitness landscape belongs to a specific problem. Unfortunately, typical fitness landscape is not smooth. In contrast, this is rugged with many hills, valleys and plateaus [31]. Valleys in the fitness landscape can usually present a barrier for the evolutionary search process that can often get stuck in local optima by experiencing them. In order to avoid this undesirable phenomenon, a lot of mechanisms have been proposed in evolutionary computation community.

Ensemble DE strategies enables an adaptation of an evolutionary search space to a fitness landscape by changing the DE strategies during the evolutionary search process. As it is known, different strategies have a different exploration power. Some of these are more exploratory, while the other focus on the exploitation of yet discovered good solutions. For instance, the 'DE/Rand/1/bin' strategy is more exploratory, while the 'DE/Best/1/bin' is more exploitative. In our study, the following strategies are taken into consideration:

1. 'DE/RandToBest/2/bin',
2. 'DE/Rand/2/bin' and
3. 'DE/CurrToRand/1'.

The first DE mutation strategy is expressed as

$$\mathbf{u}_i^{(t)} = \mathbf{x}_i^{(t)} + F(\mathbf{x}_{best}^{(t)} - \mathbf{x}_i^{(t)}) + F(\mathbf{x}_{r1}^{(t)} - \mathbf{x}_{r2}^{(t)}) + F(\mathbf{x}_{r3}^{(t)} - \mathbf{x}_{r4}^{(t)}), \tag{11}$$

where $r1$–$r4$ are randomly selected vectors from the interval $[0, Np-1]$ and $r1 \neq r2 \neq r3 \neq r4 \neq i$. The second DE mutation strategy is described as

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r1}^{(t)} + F(\mathbf{x}_{r1}^{(t)} + \mathbf{x}_{r2}^{(t)}) - F(\mathbf{x}_{r3}^{(t)} - \mathbf{x}_{r4}^{(t)} - \mathbf{x}_{r5}^{(t)}), \tag{12}$$

where $r1$–$r5$ are randomly selected vectors from the interval $[0, Np-1]$ and $r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i$. Finally, the third DE mutation strategy is defined as

$$\mathbf{u}_i^{(t)} = \mathbf{x}_i^{(t)} + F(\mathbf{x}_{best}^{(t)} - \mathbf{x}_i^{(t)}) + Q(\mathbf{x}_{r1}^{(t)} - \mathbf{x}_{r2}^{(t)}) + F(\mathbf{x}_{r3}^{(t)} - \mathbf{x}_{r4}^{(t)}), \tag{13}$$

where $r1$–$r4$ are randomly selected vectors from the interval $[0, Np-1]$, $Q \in [0, 1]$ is the value randomly drawn from uniform distribution and $r1 \neq r2 \neq r3 \neq r4 \neq i$.

The first DE mutation strategy focuses on the exploitation of the best solution, while the second on the exploration of the search space. The third DE strategy is rotationally invariant and therefore suitable for solving the rotated problems [30]. Interestingly, this strategy does not use a crossover (i.e., $CR = 1.0$) as well. The last strategy in an ensemble DE strategies can be selected optionally from the set of remaining DE strategies. In our case, one of both arithmetic crossover operators can also be selected for this optional strategy.

Additionally to changing the DE mutation strategies during an evolutionary run, changing parameters is also performed in ensemble DE algorithm (EDE). In this study, the DE control parameters, like F and CR are added to a representation of solution in EDE together with a DE mutation strategy. As a result, the solution vector is represented as

$$\mathbf{x}_i^{(t)} = [x_{i,1}, \ldots, x_{i,D}, F_i, CR_i, ST_i]^{\mathrm{T}}, \tag{14}$$

where each element $\{x_{i,j}\}$ for $i = 1, \ldots, D$ denotes a problem variable, while $F_i \in [0.1, 1.0]$, $CR_i \in [0.0, 1.0]$ and $ST_i \in [1, N_s]$ the control parameters. The parameter N_s denotes the number of different strategies in EDE, i.e., this number is fixed to $N_s = 4$ in our case. Let us notice that the control parameters F_i and CR_i are self-adapted similarly as proposed by Brest et al. [8], i.e.,

$$F_i^{(t+1))} = \begin{cases} r_1 \cdot F_{max} + F_{min}, & \text{if } r_2 < \tau_1, \\ F_i^{(t+1)}, & \text{otherwise}, \end{cases} \tag{15}$$

where F_{min} and F_{max} represent the minimum and maximum values of $F_i^{(t+1)}$, and

$$CR_i^{(t+1)} = \begin{cases} r_3, & \text{if } r_4 < \tau_2, \\ CR_i^{(t+1)}, & \text{otherwise.} \end{cases} \tag{16}$$

In Eqs. (15) and (16), r_1–r_4 denote a random number drawn from a universal distribution in the interval [0.0, 1.0], and τ_1 and τ_2 are so called learning rates. On the other hand, the DE mutation strategy ST_i is modified according to the following equation

$$ST_i^{(t+1)} = \begin{cases} (int)(r_5 \cdot N_s), & \text{if } r_6 < \tau_3, \\ ST_i^{(t)}, & \text{otherwise,} \end{cases} \tag{17}$$

where r_5 and r_6 are random numbers drawn from the universal distribution in the interval [0.0, 1.0], N_s is the number of strategies in EDE and τ_3 is a learning rate.

4. Experiments and results

The purpose of the experimental work was to show: (1) importance of the arithmetic crossover strategy in the ensemble DE strategies, and (2) additional improvement of the results by introducing the epistatic arithmetic crossover strategy. In line with this, the proposed arithmetic crossover strategies are taken into consideration in the tests. Four experiments were conducted in order to discover an influence of the proposed features. As a result, the following tests were conducted:

- influence of a scale factor K in arithmetic crossover,
- searching for an evolution window $\delta^{(0)}$ in epistatic arithmetic crossover,
- influence of an epistatic coefficient E in epistatic arithmetic crossover,
- influence of epistatic arithmetic crossover on different DE variants.

The parameter setting of the ensemble DE used in tests were the same during the tests, i.e., $F^{(0)} = 0.5$ and $CR^{(0)} = 0.9$. Each variant of these algorithms used the same population size, i.e., $Np = 100$. A termination condition represented the maximum number of fitness function evaluations and it is prescribed by the CEC-14 benchmark suite as $MAX\_FE = 10,000 \cdot D$. The total number of independent runs was set to 51. The quality of results was measured by five measures as follows:

- the minimum value achieved during 51 independent runs,
- the maximum value achieved during 51 independent runs,
- the average value achieved during 51 independent runs,
- the median value achieved during 51 independent runs,
- the standard deviation achieved during 51 independent runs.

All algorithms were applied for solving the CEC-14 benchmark function suite. The functions of dimensions $D = 10$, $D = 30$ and $D = 50$ are taken into consideration. In the remainder of paper, the CEC-14 function benchmark suite is described. Then, the proposed four experiments are described.

4.1. PC configuration

All runs were made on a computer, with the following configuration:

- Processor: Intel(R) Core(TM) i5-3470 @ 3.20 GHz,
- RAM: 8GB,
- Operating system: Linux Ubuntu 14.04.2 LTS.

All tested algorithms were implemented in the C++ programming language.

4.2. Test suite

The CEC-2014 test suite (Table 1) consists of 30 benchmark functions that are divided into four classes:

- unimodal functions (1–3),
- simple multi-modal functions (4–16),
- hybrid functions (17–22),
- composition functions (23–30).

Unimodal functions have a single global optimum and no local optimums. The unimodal functions in this suite are non-separable and rotated. Multi-modal functions are either separable or non-separable. In addition, they are also rotated or/and shifted. To develop the hybrid functions, the variables are randomly divided into some subcomponents and then different basic functions are used for different subcomponents [44]. Composition functions consist of a sum of two or more basic functions. In this suite, hybrid functions are used as the basic functions to construct composition functions. The characteristics of these hybrid and composition functions depend on the characteristics of the basic functions.

Table 1
Summary of the CEC'14 test functions.

Subgroup	No.	Functions	$F_i^* = F_i(x^*)$
Unimodal functions	1	Rotated high conditioned elliptic function	100
	2	Rotated bent Cigar function	200
	3	Rotated discus function	300
Simple multimodal functions	4	Shifted and rotated Rosenbrocks function	400
	5	Shifted and rotated Ackley's function	500
	6	Shifted and rotated Weierstrass function	600
	7	Shifted and rotated Griewanks function	700
	8	Shifted Rastrigin's function	800
	9	Shifted and rotated Rastrigins function	900
	10	Shifted Schwefels function	1000
	11	Shifted and rotated Schwefel's function	1100
	12	Shifted and rotated Katsuura function	1200
	13	Shifted and rotated HappyCat function	1300
	14	Shifted and rotated HGBat function	1400
	15	Shifted and rotated expanded Griewanks plus Rosenbrocks function	1500
	16	Shifted and rotated expanded Scaffer's F6 function	1600
Hybrid functions	17	Hybrid function 1 ($N = 3$)	1700
	18	Hybrid function 2 ($N = 3$)	1800
	19	Hybrid function 3 ($N = 4$)	1900
	20	Hybrid function 4 ($N = 4$)	2000
	21	Hybrid function 5 ($N = 5$)	2100
	22	Hybrid function 6 ($N = 5$)	2200
Composition functions	23	Composition function 1 ($N = 5$)	2300
	24	Composition function 2 ($N = 3$)	2400
	25	Composition function 3 ($N = 3$)	2500
	26	Composition function 4 ($N = 5$)	2600
	27	Composition function 5 ($N = 5$)	2700
	28	Composition function 6 ($N = 5$)	2800
	29	Composition function 7 ($N = 3$)	2900
	30	Composition function 8 ($N = 3$)	3000

4.3. *Influence of a scale factor* K *in arithmetic crossover*

The goal of this experiment was to discover an influence of a scale factor K in arithmetic crossover. This factor determines the location of the difference between elements of two randomly selected vectors according to Eq. (7). This difference can be located in the interval $[x_{r1,j}, x_{r2,j}]$ when $0 \leq K \leq 1$, below the value $x_{r1,j}$ when $K < 0$, and above the value $x_{r2,j}$ when $K > 1$. On the other hand, the value K determines the size of the search space. In order to determine the proper size, the scale factor was varied from $K = 0$ and $K = 1$ to $K = -1$ and $K = 2$ in steps of -0.1 and $+0.1$, respectively. As a result, eleven instances of the optimization problem are achieved, where the search space is gradually expanding. Thus, the size of the search space is searched for, where the best results are obtained. In line with this, the ensemble DE with arithmetic crossover (XEDE) was employed in the experiment.

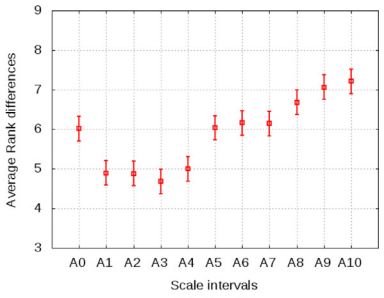
In this study, we are focused on the quality of cumulative results more than the detailed ones. Therefore, Friedman tests [49] were conducted in order to estimate the obtained results statistically. The Friedman test is a two-way analysis of variances by ranks, where the test statistic is calculated and converted to ranks in the first step, and after that the post-hoc tests are conducted using the calculated ranks in the second step. Here, a low value of rank means a better algorithm [45]. The second step is performed only if a null hypothesis of Friedman test is rejected. Note, the null hypothesis states that medians between the ranks of all algorithms are equal.

According to Demšar [48], the Friedman test is more safe and robust non-parametric test for the comparisons of more algorithms over multiple classifiers (also datasets) that together with the corresponding Nemenyi post-hoc test enables a neat presentation of statistical results [46]. The main drawback of the Friedman test is that it makes the whole multiple comparisons over data sets and therefore it is unable to establish proper comparisons between some of the algorithms considered [45]. Consequently, a Wilcoxon two paired non-parametric test is applied as a post-hoc test after determining the control method (i.e., the algorithm with the lowest rank) by using the Friedman test. Unfortunately, the Nemenyi test is very conservative and it may not find any difference in most of the experimentations [47]. Therefore, the Nemenyi test is used for graphical presentation of the results, while the Wilcoxon test shows which of the algorithms in test are more powerful. Both tests were conducted using a significance level 0.05 in this study.

The tests captured the results of optimizing all the 30 functions in the CEC-14 test suite according to all observed dimensions. In summary, eleven classifiers were compared according to $30 * 5 = 150$ variables obtained after 51 independent runs, where the first number in the expression denotes the number of functions and the second denotes the number of measures taken into consideration. In totally, the $11 * 51 = 561$ independent runs for each observed dimensions (i.e., $3 * 561 = 1683$)

K	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
[0.0,1.0]	6.02	[5.40,6.64]	†	0.00002065	†
[-0.1,1.1]	4.90	[4.28,5.52]		0.473	
[-0.2,1.2]	4.88	[4.26,5.50]		0.2622	
[-0.3,1.3]	4.68	[4.06,5.30]	‡	∞	‡
[-0.4,1.4]	5.00	[4.38,5.62]		0.7063	
[-0.5,1.5]	6.04	[5.42,6.66]	†	0.0217	†
[-0.6,1.6]	6.16	[5.54,6.78]	†	0.07424	
[-0.7,1.7]	6.15	[5.53,6.77]	†	0.07149	
[-0.8,1.8]	6.68	[6.06,7.30]	†	0.01059	†
[-0.9,1.9]	7.06	[6.44,7.68]	†	0.005386	†
[-1.0,2.0]	7.21	[6.59,7.83]	†	0.01159	†

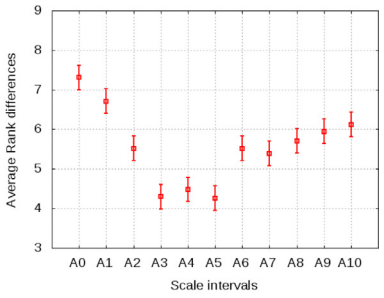
(a) $D = 10$



(b) $D = 10$

K	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
[0.0,1.0]	7.31	[6.69,7.93]	†	9.358E-008	†
[-0.1,1.1]	6.71	[6.09,7.33]	†	1.269E-007	†
[-0.2,1.2]	5.52	[4.90,6.14]	†	0.1437	†
[-0.3,1.3]	4.3	[3.68,4.92]	‡	∞	‡
[-0.4,1.4]	4.48	[3.86,5.10]		0.5519	
[-0.5,1.5]	4.26	[3.64,4.88]		0.4343	
[-0.6,1.6]	5.52	[4.90,6.14]	†	0.000195	†
[-0.7,1.7]	5.39	[4.77,6.01]		0.0037	†
[-0.8,1.8]	5.71	[5.09,6.33]	†	0.0001971	†
[-0.9,1.9]	5.95	[5.33,6.57]	†	0.0004994	†
[-1.0,2.0]	6.12	[5.50,6.74]	†	0.00002228	†

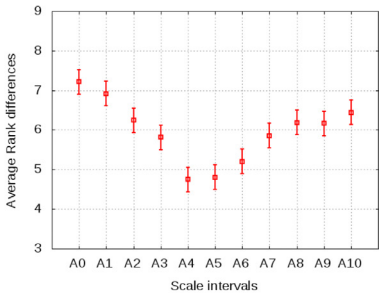
(c) $D = 30$



(d) $D = 30$

K	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
[0.0,1.0]	7.21	[6.59,7.83]	†	2.90E-005	†
[-0.1,1.1]	6.92	[6.30,7.54]	†	0.0001242	†
[-0.2,1.2]	6.24	[5.62,6.86]	†	0.003981	†
[-0.3,1.3]	5.81	[5.19,6.43]		0.002899	†
[-0.4,1.4]	4.75	[4.13,5.37]	‡	∞	‡
[-0.5,1.5]	4.8	[4.18,5.42]		0.3237	
[-0.6,1.6]	5.2	[4.58,5.82]		0.1437	
[-0.7,1.7]	5.85	[5.23,6.47]		0.07809	
[-0.8,1.8]	6.19	[5.57,6.81]	†	0.01512	†
[-0.9,1.9]	6.16	[5.54,6.78]	†	0.05513	
[-1.0,2.0]	6.44	[5.82,7.06]	†	0.008055	†

(e) $D = 50$



(f) $D = 50$

Fig. 3. Influence of the scale factor K in arithmetic crossover.

must be conducted in order to publish the results presented in Fig. 3. The figure is divided to three parts according to the observed dimensions, where each part consists of a table denoting the results of three statistical tests (i.e., Friedman, Nemenyi and Wilcoxon), and a diagram presenting the results of Nemenyi test graphically. In diagrams, labels A0–A10 denote the intervals from $K \in [0.0, 1.0]$ (standard DE) to $K \in [-1.0, 2.0]$ in steps of 0.1.

From Fig. 3 it can be seen that the interval limiting the search space is increased with increasing the dimension of the problem. This means that the optimal values of $K \in [-0.3, 1.3]$ were found for dimensions $D = 10$ and $D = 30$, while for dimension $D = 50$, the best values of K lay in the interval $K \in [-0.4, 1.4]$.

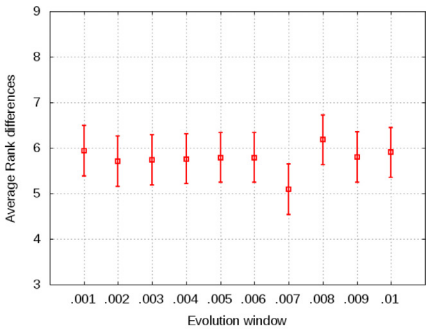
4.4. Searching for an evolution window $\delta^{(0)}$ in epistatic arithmetic crossover

An evolution window is characteristics of the normal mutation that indicates an order of magnitude within the reasonable performance is observed [16]. This is controlled by the starting values of dependence vectors $\delta_{i,j}$ for $i = 1, \ldots, Np \wedge j = 1, \ldots, D$. In line with this, the starting values of these vectors were varied in the interval [0.001, 0.010] in steps of 0.001. Thus, ten instances of the optimization problem were obtained. In summary, the $10 * 51 = 510$ independent runs for each of three observed dimensions (i.e., $3 * 510 = 1,530$) were conducted. The cumulative results of the optimization are presented in Fig. 4 that is organized similar as those illustrated in the last section.

As can be seen from Fig. 4, the selection of the starting values of dependence vectors $\delta_{i,j}$ has a great influence on the results of the optimization regarding the Wilcoxon tests. The best results were obtained by initialization of dependence vectors by $\delta_{i,j} = 0.007$ for dimension $D = 10$, by $\delta_{i,j} = 0.004$ for dimension $D = 30$, and by $\delta_{i,j} = 0.007$ for dimension $D = 50$. According to Wilcoxon signed rank post-hoc test, these values significantly outperformed the majority of the other observed instances of dimensions $D = 10$ and $D = 30$, while the difference between instances was not significant by dimension $D = 50$.

$\delta^{(0)}$	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
0.001	5.94	[5.39,6.49]		0.0001166	†
0.002	5.71	[5.16,6.26]		0.007272	†
0.003	5.74	[5.19,6.29]		0.007728	†
0.004	5.76	[5.21,6.31]		0.02265	†
0.005	5.79	[5.24,6.34]		0.114	
0.006	5.79	[5.24,6.34]		0.06683	
0.007	5.09	[4.54,5.64]	‡	∞	‡
0.008	6.18	[5.63,6.73]		0.000583	†
0.009	5.80	[5.25,6.35]		0.05611	
0.010	5.90	[5.35,6.45]		0.02323	†

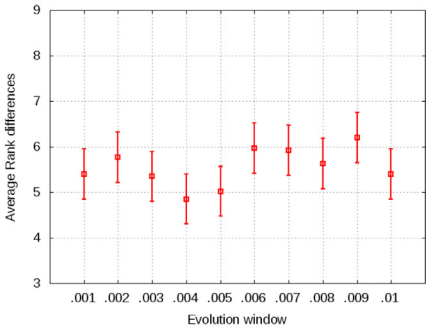
(a) $D = 10$



(b) $D = 10$

$\delta^{(0)}$	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
0.001	5.40	[4.85,5.95]		0.004948	†
0.002	5.77	[5.22,6.32]		0.02833	†
0.003	5.35	[4.80,5.90]		0.01503	†
0.004	4.85	[4.30,5.40]	‡	∞	‡
0.005	5.02	[4.47,5.57]		0.6799	
0.006	5.97	[5.42,6.52]	†	0.000324	†
0.007	5.92	[5.37,6.47]		7.25E-005	†
0.008	5.63	[5.08,6.18]		0.025	†
0.009	6.20	[5.65,6.75]	†	0.0002208	†
0.010	5.40	[4.85,5.95]		0.004948	†

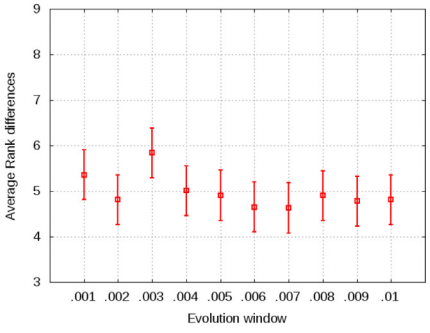
(c) $D = 30$



(d) $D = 30$

$\delta^{(0)}$	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
0.001	5.36	[4.81,5.91]		0.08672	
0.002	4.81	[4.26,5.36]		0.09932	
0.003	5.84	[5.29,6.39]	†	0.07067	
0.004	5.01	[4.46,5.56]		0.2138	
0.005	4.91	[4.36,5.46]		0.2859	
0.006	4.65	[4.10,5.20]		0.3955	
0.007	4.63	[4.08,5.18]	‡	∞	‡
0.008	4.90	[4.35,5.45]		0.3303	
0.009	4.78	[4.23,5.33]		0.8093	
0.010	4.81	[4.26,5.36]		0.2148	

(e) $D = 50$



(f) $D = 50$

Fig. 4. Evolutionary window in epistatic arithmetic crossover.

4.5. *Influence of an epistatic coefficient* E *in the arithmetic crossover*

An epistatic coefficient E regulates the influence of the epistatic elements of vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ on the trial vector. This influence is controlled by an epistatic coefficient E in the epistatic arithmetic crossover. The higher the value, the stronger the influence of the second parent. In order to discover what impact this parameter has on the results of the optimization, this was varied in the interval $E \in [0.0, 1.0]$ in steps of 0.1. As a result, eleven instances of the optimization problem were obtained. In line with this, the $11 * 51 = 561$ independent runs for each of three observed dimensions (i.e., $3 * 561 = 1,683$) were executed.

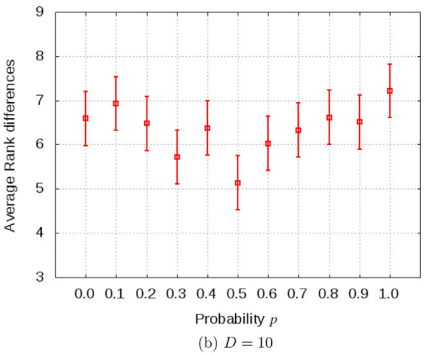
As can be seen from Fig. 5, the best results were obtained by $E = 0.5$ by each observed dimension. Mainly, this instance was also significantly better as the other instances according to Wilcoxon signed rank statistical test. In summary, equivalent impact of epistatic elements in both parents produce the best results.

4.6. *Comparative analysis*

The purpose of this experiment was to show that using the arithmetic crossover in ensemble DE strategies can outperform the results of the other DE variants, like the original DE [7], and self-adaptive variants jDE [8] and SaDE [9]. In line with this, three ensemble DE strategy algorithms have been developed, i.e., ensemble DE using the scale factor K (EDE), ensemble DE with arithmetic crossover strategy (XEDE) and ensemble DE with the epistatic arithmetic crossover strategy (eXEDE). Let us notice that the SaDE algorithm beside the adaptation of a scale factor F and a crossover rate CR in DE mutation strategies,

E	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
0.0	6.59	[5.97,7.21]	†	0.0006497	†
0.1	6.93	[6.31,7.55]	†	0.00003064	†
0.2	6.48	[5.86,7.10]	†	0.0006004	†
0.3	5.72	[5.10,6.34]		0.05432	
0.4	6.38	[5.76,7.00]		0.00001942	†
0.5	5.14	[4.52,5.76]	‡	∞	‡
0.6	6.03	[5.41,6.65]		0.009609	†
0.7	6.33	[5.71,6.95]		0.01317	†
0.8	6.62	[6.00,7.24]	†	0.0003493	†
0.9	6.51	[5.89,7.13]	†	0.001674	†
1.0	7.22	[6.60,7.84]	†	0.000002354	†

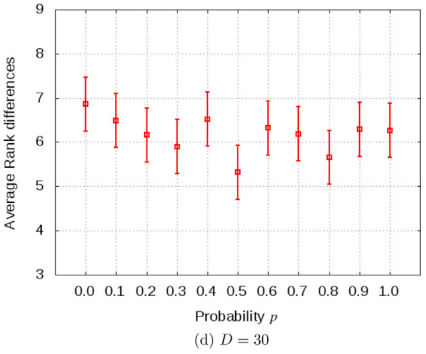
(a) $D = 10$



(b) $D = 10$

E	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
0.0	6.86	[6.24,7.48]		0.06882	
0.1	6.49	[5.87,7.11]		0.2498	
0.2	6.16	[5.54,6.78]		0.03985	†
0.3	5.90	[5.28,6.52]		0.4358	
0.4	6.52	[5.90,7.14]		0.02597	†
0.5	5.32	[4.70,5.94]	‡	∞	‡
0.6	6.32	[5.70,6.94]		0.02533	†
0.7	6.19	[5.57,6.81]		0.5371	
0.8	5.66	[5.04,6.28]		0.8527	
0.9	6.29	[5.67,6.91]		0.01749	†
1.0	6.27	[5.65,6.89]		0.06586	†

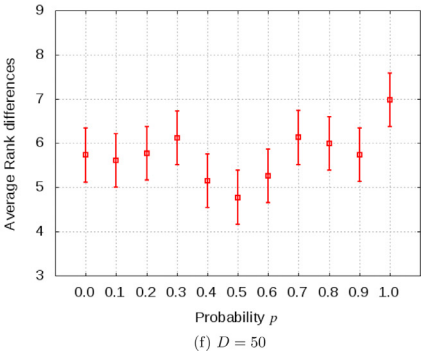
(c) $D = 30$



(d) $D = 30$

E	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
0.0	5.73	[5.11,6.35]		0.09282	
0.1	5.61	[4.99,6.23]		0.1168	
0.2	5.77	[5.15,6.39]		0.449	
0.3	6.12	[5.50,6.74]	†	0.01912	†
0.4	5.15	[4.53,5.77]		0.4726	
0.5	4.77	[4.15,5.39]	‡	∞	‡
0.6	5.26	[4.64,5.88]		0.6836	
0.7	6.13	[5.51,6.75]	†	0.002141	†
0.8	5.99	[5.37,6.61]		0.008472	†
0.9	5.74	[5.12,6.36]		0.05043	
1.0	6.98	[6.36,7.60]	†	0.0001043	†

(e) $D = 50$



(f) $D = 50$

Fig. 5. Epistatic coefficient E in arithmetic crossover.

adapts also type of DE mutation strategy in order to modify a kind of exploring the search space. A difference between both algorithms is that the SaDE selects the new strategy with roulette wheel selection, while the EDE uses Eq. (17) for changing the type of strategy.

The following parameter setting was used during the experiments. The scaling factor $F = 0.5$ and crossover rate $CR = 0.9$ were applied for the DE algorithm. The same initial values were used for jDE. In our SaDE implementation, the following mutation strategies were applied: 'rand/1/bin', 'rand/2/bin', 'rand-to-best/2/bin' and 'current-to-rand/1'. The learning period was set as $LP = 20$, while the scale parameters F_i were changed using the normal distribution $N(0.5, 0.3)$ with mean value 0.5 and standard deviation 0.3, and crossover rates CR_i were randomly drawn from normal distribution $N(CRm_k, 0.1)$ with mean value CRm_k and standard deviation 0.1.

The results of the optimization of CEC-14 benchmark suite by solving dimension of functions $D = 30$ are illustrated in Table 2. Due to the paper length limitation, the detailed results of the other dimensions of the functions (i.e., $D = 10$ and $D) = 30$) are omitted. Note that the best results as achieved by the optimization are presented bold.

As can be seen from Table 2, the original DE achieved the best results three times, jDE once, SaDE five times. EDE four times, XEDE eight times and eXEDE seven times. In summary, ensemble DE using arithmetic and epistatic arithmetic crossover (XEDE and eXEDE) outperformed the other algorithms in test substantially.

The cumulative results of the optimizing the CEC-14 benchmark functions according to all observed dimensions (i.e., $D = 10$, $D = 30$ and $D = 50$) are presented in Table 6.

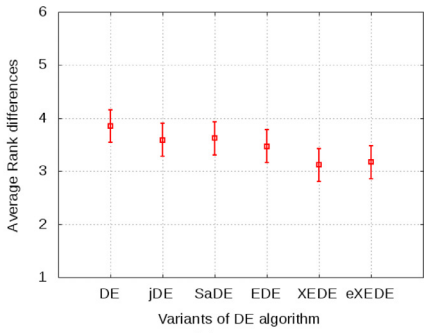
Table 2
Comparison of mean and std values for algorithms used in the study on dimension $D = 30$.

F	Meas.	DE	jDE	SaDE	EDE	XEDE	eXEDE
f_1	Mean	101465	61174.6	3725.56	78881.4	5638.95	**3997.18**
	Std	89819.5	76381	3261.35	61701.6	4513.23	4065.14
f_2	Mean	2.27E–015	2.27E–015	1.71E–014	**1.14E–015**	2.39E–014	2.39E–014
	Std	7.87E–015	7.87E–015	1.47E–014	5.68E–015	1.06E–014	1.06E–014
f_3	Mean	**2.05E–014**	4.09E–014	6.25E–014	2.50E–014	5.46E–014	5.68E–014
	Std	2.78E–014	2.60E–014	1.80E–014	2.88E–014	1.14E–014	0
f_4	Mean	2.84123	8.53396	**1.53E–013**	1.16E+001	1.68E–013	1.77E–013
	Std	12.6172	21.6167	7.11E–014	24.6342	9.21E–014	6.21E–014
f_5	Mean	20.8904	20.3361	**20.289**	20.3682	20.4061	20.3889
	Std	0.0767006	0.0325836	0.0403143	0.0345644	0.0465178	0.0305861
f_6	Mean	4.11751	5.30804	14.9301	6.71426	**1.37047**	3.69622
	Std	3.10541	4.03538	0.941979	4.69168	1.76997	1.63413
f_7	Mean	0.000295842	0.000295842	9.09E–014	**8.64E–014**	0.000295842	0.00373993
	Std	0.00147921	0.00147921	4.79E–014	4.96E–014	0.00147921	0.0079714
f_8	Mean	65.2201	0.119395	1.02E–013	0	**4.55E–015**	0.278589
	Std	31.7165	0.329991	3.60E–014	0	2.27E-014	0.886203
f_9	Mean	173.543	38.0751	35.8277	46.2976	29.5789	**25.9375**
	Std	10.8376	5.71342	7.00529	6.14563	5.17458	5.29906
f_{10}	Mean	2144.97	3.17197	**1.10508**	2.42121	20.7329	10.2815
	Std	980.44	3.18171	2.01974	2.91087	3.15085	6.29153
f_{11}	Mean	6699.81	2707.35	2278.88	2738.17	2643.17	**2331.92**
	Std	323.601	274.56	344.643	249.438	341.429	317.283
f_{12}	Mean	2.39944	0.476899	**0.458619**	0.51757	0.517386	0.508992
	Std	0.297368	0.0541086	0.0523364	0.0665004	0.0766053	0.0763449
f_{13}	Mean	0.317998	0.284315	0.301583	0.290451	0.241216	**0.210002**
	Std	0.0429744	0.0355372	0.0369317	0.0354088	0.0328373	0.0360487
f_{14}	Mean	0.27284	0.302073	0.267787	0.298374	0.270534	**0.237317**
	Std	0.0306132	0.0414956	0.139632	0.0281479	0.0436366	0.0354316
f_{15}	Mean	14.8258	5.36367	4.86126	5.64289	4.62401	**4.02045**
	Std	1.12613	0.742653	0.41658	0.653411	0.727968	0.708776
f_{16}	Mean	12.5109	10.2964	10.3215	10.5327	10.2855	**9.90624**
	Std	0.240958	0.323022	0.342	0.231708	0.442214	0.472961
f_{17}	Mean	1283.14	1624.38	855.477	1809.15	**764.487**	803.527
	Std	340.352	1485.54	280.061	2548.08	313.763	247.283
f_{18}	Mean	50.8323	**18.565**	49.1882	20.1481	47.9372	50.2522
	Std	16.59	10.3504	25.6845	10.7412	23.9881	21.1402
f_{19}	Mean	4.88768	4.96616	5.259	5.25006	**4.67343**	5.36851
	Std	0.858623	0.961318	1.15292	0.720126	0.654865	0.761403
f_{20}	Mean	12.4475	13.5782	18.4514	**10.78**	15.2427	17.2232
	Std	6.76961	6.63615	4.13879	3.39675	4.85249	4.63811
f_{21}	Mean	275.343	297.879	431.453	**203.32**	321.316	337.28
	Std	253.179	224.848	131.92	179.125	110.081	121.683
f_{22}	Mean	121.209	137.556	164.526	121.524	124.561	**120.862**
	Std	122.111	53.7631	71.0822	51.3297	65.5073	69.8855
f_{23}	Mean	315.244	**315.244**	**315.244**	**315.244**	**315.244**	**315.244**
	Std	9.28E–014	0	0	0	0	0
f_{24}	Mean	**222.362**	226.488	225.169	225.563	225.006	225.152
	Std	7.05938	3.33795	4.31446	1.77416	2.17037	1.47548
f_{25}	Mean	**202.76**	203.553	203.458	203.391	203.494	203.226
	Std	0.219477	0.880935	0.551686	0.655508	0.921585	0.571736
f_{26}	Mean	100.316	100.276	100.302	100.298	100.266	**100.221**
	Std	0.0418593	0.0505395	0.0354606	0.0401162	0.0470502	0.0372353
f_{27}	Mean	378.338	401.426	546.126	391.417	**325.979**	379.194
	Std	82.2668	54.3829	110.996	31.0188	30.9886	29.7201
f_{28}	Mean	843.542	838.246	**808.137**	823.568	820.632	919.525
	Std	47.3228	29.8885	37.7788	24.3492	32.5525	179.476
f_{29}	Mean	682617	865.619	840897	820.335	**723.047**	1732130
	Std	2.36E+006	161.798	2.66E+006	100.367	41.3626	3535000
f_{30}	Mean	1956.74	2788.4	2342.55	2471.52	**1543.2**	1865.84
	Std	1241.54	1216.2	1380.57	1000.43	716.631	1096.27

From Fig. 6, it can be observed that using the arithmetic crossover in XEDE significantly outperformed the results of the other DE variants (i.e., DE and SaDE) by optimizing the CEC-14 benchmark functions of dimension $D = 10$. The same variants of the DE algorithm was significantly better that all the other algorithms except eXEDE by optimizing the CEC-14 benchmark functions of dimension $D = 30$, while the eXEDE was the best algorithm by optimizing the functions of the higher dimensions (i.e., $D = 50$). This means that the epistatic arithmetic crossover works well by exploring the higher search spaces.

Algs.	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
DE	3.85	[3.54,4.16]	†	0.001627	†
jDE	3.59	[3.28,3.90]		0.07464	
SaDE	3.62	[3.31,3.93]		0.00006192	†
EDE	3.47	[3.16,3.78]		0.3365	
XEDE	3.12	[2.81,3.43]	‡	∞	‡
eXEDE	3.17	[2.86,3.48]		0.6132	

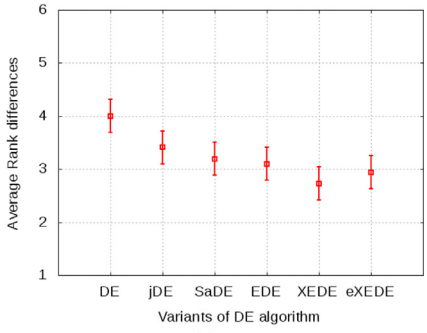
(a) $D = 10$



(b) $D = 10$

Algs.	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
DE	4.04	[3.73,4.35]	†	1.67E-011	†
jDE	3.45	[3.14,3.76]	†	2.75E-006	†
SaDE	3.23	[2.92,3.54]		0.005578	†
EDE	3.17	[2.86,3.48]		0.004189	†
XEDE	2.69	[2.38,3.00]	‡	∞	‡
eXEDE	2.80	[2.49,3.11]		0.575	

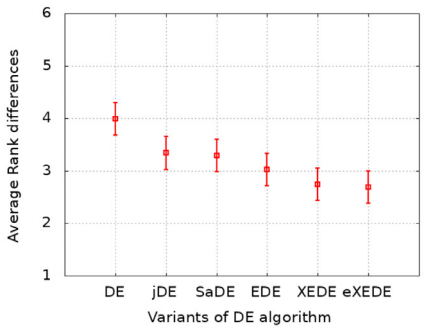
(c) $D = 30$



(d) $D = 30$

Algs.	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
DE	3.95	[3.64,4.26]	†	9.881e-07	†
jDE	3.29	[2.98,3.60]		0.002869	†
SaDE	3.24	[2.93,3.55]		0.007039	†
EDE	2.89	[2.58,3.20]		0.2836	
XEDE	3.03	[2.72,3.34]		0.003682	†
eXEDE	2.74	[2.43,3.05]	‡	∞	‡

(e) $D = 50$



(f) $D = 50$

Fig. 6. Influence of the arithmetic crossover on DE algorithms.

5. Conclusion

This paper deals with an impact of arithmetic crossover in the ensemble DE strategies by solving the CEC-14 benchmark suite. Beside the pure arithmetic crossover, the so called epistatic arithmetic crossover was proposed that also takes into consideration an impact of epistatic genes. In genetics, more than one gene has an impact on specific traits, in general. This influence is handled in genetics as epistasis. Here, the epistasis is expressed as a graph product of two linear graphs determined by vectors (i.e., candidate solutions) that enter into crossover operation. Although there are many graph products we focus on the Cartesian graph product only in the study.

In the experimental work, an influence of the arithmetic crossover on the results of optimization was discovered. In line with this, three tests were performed in which the influence of the scale factor K in arithmetic crossover was verified, an evolution window $\delta^{(0)}$ in epistatic arithmetic crossover was searched for, and influence of an epistatic coefficient E in epistatic arithmetic crossover was discovered. Finally, the ensemble DE strategies using arithmetic crossover (i.e., EDE, XEDE and eXEDE) were compared with the original DE and two self-adaptive DE variants, i.e., jDE and SaDE. This comparative study showed an appropriateness of both arithmetic strategies because using the arithmetic and epistatic arithmetic crossover within ensemble DE strategies significantly outperformed the results of the other algorithms in tests.

Epistatic arithmetic crossover demonstrated the best results especially by solving the CEC-14 benchmark functions of higher dimensions. As a future work, we would like to study Cartesian product of graphs which are not necessarily isomorphic to linear graphs. Moreover, we could apply also other graph products, like strong, direct and lexicographic, in the epistatic arithmetic crossover.

References

[1] S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Trans. Evolut. Comput. 15 (1) (2011) 4–31.
[2] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Natural Computing Series, Springer, Verlag, Berlin, 2003.
[3] Y. Davidor, Epistasis variance: Suitability of a representation to genetic algorithms, Complex Syst. 4 (1990) 369–383.
[4] Y. Davidor, Epistasis variance: A viewpoint on GA-hardness, Found. Genet. Algorithms 1 (1991) 23–35.
[5] R. Gras, How efficient are genetic algorithms to solve high epistasis deceptive problems? IEEE Congr. Evolut. Comput. (2008) 242–249.
[6] S.A. Kauffman, E.D. Weinberger, The NK model of rugged fitness landscapes and its application to maturation of the immune response, J.Theor. Biol. 141 (2) (1989) 211–245.
[7] R. Storn, K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[8] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark, IEEE Trans. Evolut. Comput. 10 (6) (2006) 646–657.
[9] K.A. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evolut. Comput. 13 (2) (2009) 398–417.
[10] K.A. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, IEEE Congr. Evolut. Comput. 2 (2005) 1785–1791.
[11] R. Mallipeddi, S. Mallipeddi, P.N. Suganthan, Ensemble strategies with adaptive evolutionary programming, Inf. Sci. 180 (9) (2010) 1571–1581.
[12] R. Mallipeddi, P.N. Suganthan, Q.-K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.
[13] J. Tvrdìk, Differential evolution with competitive setting of its control parameters, TASK Quarterly 11 (2007) 169–179.
[14] J. Tvrdìk, Adaptation in differential evolution: A numerical comparison, Appl. Soft Comput. 9 (3) (2009) 1149–1155.
[15] C. Darwin, On the Origin of Species, Harvard University Press, London, UK, 1859.
[16] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, Oxford, UK, 1996.
[17] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, MA, 1996.
[18] J. Koza, Genetic Programming 2 - Automatic Discovery of Reusable Programs, MIT Press, Cambridge, USA, 1994.
[19] L. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence through Simulated Evolution, John Willey, New York, US, 1966.
[20] S. Hui, P.N. Suganthan, Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization, IEEE Trans. Cybern. 99 (2015) 1–11, doi:10.1109/TCYB.2015.2394466.
[21] A.H. Wright, Genetic algorithms for real parameter optimization, in: Foundations of Genetic Algorithms, Morgan Kaufmann, 1991, pp. 205–218.
[22] F. Neri, V. Tirronen, Recent advances in differential evolution: A survey and experimental analysis, Artif. Intell. Rev. 33 (1–2) (2010) 61–106.
[23] R.B. Heckendorn, A.H. Wright, Efficient linkage discovery by limited probing, in: Genetic and Evolutionary Computation, Springer Berlin Heidelberg, 2004, pp. 1003–1014.
[24] H. Matthew, Population Genetics, Wiley-Blackwell, 2009.
[25] W. Imrich, S. Klavžar, Product Graphs: Structure and Recognition, Wiley-Interscience, New York, 2000.
[26] W. Imrich, I. Peterin, Recognizing cartesian products in linear time, Discret. Math. 307 (2007) 472–483.
[27] G. Sabidussi, Graph multiplication, Math. Z. 72 (1) (1960) 446–457.
[28] V.G. Vizing, The cartesian product of graphs (russian), Vyčisl Sistemy 9 (1963) 30–43.
[29] S. Wright, The roles of mutation, inbreeding, crossbreeding, and selection in evolution, Proc. Sixth Int. Congr. Genet. 1 (1932) 356–366.
[30] A.W. Iorio, X. Li, Solving rotated multi-objective optimization problems using differential evolution, in: AI 2004: Advances in Artificial Intelligence, vol. 3339, Springer, Berlin Heidelberg, 2005, pp. 861–872.
[31] P.F. Stadler, Fitness landscapes, Appl. Math. Comput. 117 (2002) 187–207.
[32] V. Vukašinovič, P. Gregor, R. Škrekovski, On the mutually independent hamiltonian cycles in faulty hypercubes, Inf. Sci. 236 (2013) 224–235.
[33] P. Gregor, R. Škrekovski, V. Vukašinovič, Rooted level-disjoint partitions of cartesian products, Appl. Math. Comput. 266 (2015) 244–258.
[34] P. Gregor, R. Škrekovski, V. Vukašinovič, Queue layouts of hypercubes, SIAM J. Discret. Math. 26 (2012) 77–88.
[35] D. Dimitrov, T. Dvořák, P. Gregor, R. Škrekovski, Linear time construction of a compressed gray code, Eur. J. Comb. 34 (1) (2013) 69–81.
[36] D. Dimitrov, T. Dvořák, P. Gregor, R. Škrekovski, Gray codes avoiding matchings, Discret. Math. Theor. Comput. Sci. 11 (2) (2009) 123–148.
[37] M. Zanin, A.N. Pisarchik, Gray code permutation algorithm for high-dimensional data encryption, Inf. Sci. 270 (20) (2014) 288–297.
[38] F. Zhou, H. Toivonen, R.D. King, The use of weighted graphs for large-scale genome analysis, PLoS ONE 9 (3) (2014) e89618.
[39] A. Adebimpe, A. Aarabi, E. Bourel-Ponchel, M. Mahmoudzadeh, F. Wallois, Functional brain dysfunction in patients with benign childhood epilepsy as revealed by graph theory, PLoS ONE 10 (10) (2015) e0139228.
[40] V. Stavrakas, I.N. Melas, T. Sakellaropoulos, Alexopoulos, G. Leonidas, Network reconstruction based on proteomic data and prior knowledge of protein connectivity using graph theory, PLoS ONE 10 (5) (2015) e0128411.
[41] A. Sandryhaila, J. Moura, Big data analysis with signal processing on graphs, IEEE Signal Process. Mag. 31 (5) (2014) 80–90.
[42] S. Klavžar, Applications of isometric embeddings to chemical graphs, DIMACS Ser. Discret. Math. Theor. Comput. Sci. 51 (2000) 249–259.
[43] M.H. Khalifeh, H. Yousefi-Azari, A.R. Ashrafi, Vertex and edge PI indices of cartesian product graphs, Discret. Appl. Math. 156 (10) (2008) 1780–1788.
[44] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Computational Intelligence Laboratory, Zhengzhou University, Nanyang Technological University, Singapore, 2013.Technical Report
[45] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (1) (2011) 3–18.
[46] P.B. Nemenyi, Distribution-Free Multiple Comparisons (Ph.D. thesis) Princeton University, 1963.
[47] S. García, F. Herrera, An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons, J. Mach. Learn. Res. 7 (2008) 2677–2694.
[48] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[49] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. Am. Stat. Assoc. 11 (1940) 86–92.
[50] P. Žigert Pleteršek, M. Berlič, Resonance graphs of armchair nanotubes cyclic polypyrenes and amalgams of Lucas cubes, MATCH Commun. Math. Comput. Chem. 70 (2013) 533–543.
[51] R. Erveš, J. Žerovnik, Improved upper bounds for vertex and edge fault diameters of cartesian graph bundles, Discret. Appl. Math. 181 (2015) 90–97.