Is Algebraic Differential Evolution Really a Differential Evolution Scheme?

Valentino Santucci
Department of Humanities and Social Sciences
University for Foreigners of Perugia
Perugia, Italy
valentino.santucci@unistrapg.it

***Abstract*—The Algebraic Differential Evolution (ADE) is a recently proposed combinatorial evolutionary scheme which mimics the behaviour of the classical Differential Evolution (DE) in discrete search spaces which can be represented as finitely generated groups. ADE has been successfully applied to both permutation and binary optimization problems. However, in the previous works, the relationship between ADE and the classical continuous DE has been only intuitively sketched without any theoretical or experimental proof. Here, we fill this gap by providing both theoretical and experimental justifications proving that ADE is a full-fledged generalization of DE which works across different search spaces. First, we formally prove that there exists a concrete implementation of ADE's algebraic operations converging to the classical vector operations of DE, then we propose a real-vector implementation of ADE and we experimentally prove that its behaviour is statistically equivalent to DE. As conclusion, we also pave the way for further applications of the original DE idea to mixed discrete/continuous search spaces.**

***Index Terms*—Differential Evolution, Algebraic Differential Evolution, Search spaces, Theoretical analysis**

I. Introduction and Related Work

The Algebraic Differential Evolution (ADE) [1] is a recently proposed combinatorial variant of the classical continuous Differential Evolution (DE) scheme [2] which obtained interesting results on some combinatorial optimization problems.

ADE is an abstract algorithmic scheme, built on the basis of solid group theory concepts [3], which can be instantiated for different combinatorial spaces and representations such as permutations and binary strings.

In [4] and [5], ADE has been applied to the Permutation Flowshop Scheduling Problem (PFSP) obtaining, respectively, state-of-the-art results for the total flowtime criterion [4] and competitive performances for the makespan objective [5]. Another permutation-based problem successfully addressed by ADE is the Linear Ordering Problem (LOP) investigated in [6]. Then, in [7], a multi-neighborhood version of ADE obtained new state-of-the-art results on the cumulative-costs variant of the LOP. Still with regard to problems whose solutions can be represented by permutations of items, ADE has also been successfully applied to the Traveling Salesman Problem (TSP) and the Single Row Facility Layout Problem (SRFLP) in [8] and [9], respectively. Binary instantiations of ADE have been proposed in [3] and [10] for, respectively, artificial NK-Landscape benchmarks and the MultiDimensional TwoWay Number Partitioning Problem (MDTWNPP). In particular, new state-of-the-art results for the MDTWNPP have been obtained. Furthermore, an implementation of ADE for a composite genotype encoding, formed by a permutation and a bit-string, has been designed in [11] for learning the structure of Bayesian networks from data, while an adaptation for non-group spaces is provided in [12] for the representation of permutations with repetition.

Besides ADE, in literature there are other approaches for making DE work in combinatorial spaces like e.g. [13] and [14] for the binary representation and [15] for the permutations space. They work by separating the genotype and phenotype representations, i.e., a solution is represented as a real-vector and evolved by DE as usual, then, it is decoded to the discrete phenotype just before its evaluation. However, this decoder-based approach has been criticized in [3], which provides theoretical and experimental arguments showing that the commonly understood behaviour of the DE search in the continuous space is completely lost when the search progress is observed in the phenotypic discrete space.

Conversely, the ADE approach does not suffer from this issue. Nevertheless, the parallels between the search behaviours, of ADE in the discrete space and of DE in the continuous space, have only been drawn by means of intuition. Neither theoretical nor experimental proofs have been given before. Here, we fill this gap by providing both theoretical and experimental justifications proving that ADE is a full-fledged generalization of DE.

We follow this line:

1) we introduce an implementation of the algebraic framework for the space of integer vectors;
2) we show that this integer instantiation is isomorphic to a set of real vectors denoted by $\mathbb{R}_h^n$ and parameterized by a positive value $h \in \mathbb{R}^+$;
3) we introduce an implementation of ADE, namely ADE-RV, working on the real vectors of $\mathbb{R}_h^n$;
4) we theoretically prove that ADE-RV operations converge to the operations of the original DE when h tends to zero;
5) we experimentally prove that, by setting h to a small enough value, the search behaviour of ADE-RV is statistically equivalent to that of DE on a number of widely adopted benchmark problems.

This will show that ADE is an abstract scheme which generalizes and subsumes the classical DE, making it possible to apply, in a principled way, the same searching behaviour of DE across different kinds of search spaces.

The rest of the paper is organized as follows. The abstract scheme of ADE is briefly recalled in Section II, while Section III describes the algebraic framework on which the ADE operations are based. The implementation of ADE for real vectors, i.e. ADE-RV, is introduced in Section IV. Then, Section V provides a theoretical proof that ADE-RV operations are equivalent, in the limit, to the vector operations of the classical DE, while Section VI provides experimental evidences that ADE-RV behaves like its classical counterpart. Finally, conclusions are drawn in Section VII, where important future lines of research are also depicted.

II. Abstract Algebraic Differential Evolution

While the classical Differential Evolution (DE) [1] aims at optimizing a numerical objective function of the form $f : \mathbb{R}^n \to \mathbb{R}$, the Algebraic Differential Evolution (ADE) [4] tackles optimization problems of the form $f : X \to \mathbb{R}$, where the domain X is a set of discrete candidate solutions which satisfies the properties of a finitely generated group.

Examples of domains suitable for ADE are the set of n-length bit-strings $\mathbb{B}^n$ (see e.g. [10]) and the set of permutations of n items $\mathbb{S}_n$ (see e.g. [7]). Here below we provide an abstract description of the ADE scheme which is valid for any choice of the finitely generated group X.

ADE follows the same outline of DE, but its population is composed of discrete solutions from the search domain X at hand. Its pseudo-code is depicted in Alg. 1.

Algorithm 1 Abstract scheme of ADE

```
1: Randomly initialize $N$ solutions $x_1, \ldots, x_N \in X$
2: while termination criterion is not satisfied do
3:     for $i \leftarrow 1, \ldots, N$ do
4:         $y_i \leftarrow$ AlgebraicDifferentialMutation($x_i$)
5:         $z_i \leftarrow$ Crossover($x_i, y_i$)
6:     for $i \leftarrow 1, \ldots, N$ do
7:         if $f(z_i) < f(x_i)$ then          ▷ Minimization is assumed
8:             $x_i \leftarrow z_i$
9: return $x_{\text{best}}$
```

A population of N candidate solutions $x_1, \ldots, x_N \in X$ is randomly initialized in line 1. Then, in lines 2–8, the population is iteratively evolved by means of three genetic operators: algebraic differential mutation, crossover and selection. After the evolution loop, the best solution encountered is returned in line 9.

Similarly to DE, the core operator of ADE is the algebraic differential mutation (line 4) which, in its most used variant "rand/1", for every population individual x_i, generates a mutant $y_i \in X$ as follows:

$$y_i \leftarrow x_{r_0} \oplus F \odot (x_{r_1} \ominus x_{r_2}), \tag{1}$$

where: $F > 0$ is the *scale factor* parameter of ADE, while x_{r_0}, x_{r_1} and x_{r_2} are three randomly chosen population individuals different among them and with respect to x_i. Most importantly, $\oplus, \ominus, \odot$ are the algebraic operators which make the differential mutation work in the discrete space X by simulating the behaviour of their numerical counterparts in $\mathbb{R}^n$.

The definitions and the rationale behind the algebraic operators are given in Section III, while the proof that they subsume the usual vector operations of $\mathbb{R}^n$ is the object of this study.

The mutant y_i is recombined in line 5 with the corresponding population individual x_i, thus a trial solution $z_i \in X$ is generated. The recombination is performed using one of the discrete crossover operators available in the literature [16], [17] for the search space X at hand. The choice of the crossover operator is usually problem-dependent and left to the practitioner. The only requirement is that the crossover has to natively work with solutions from X. Moreover, a *crossover strength* parameter $CR \in [0, 1]$ is usually adopted as in the original DE. For instance, in [3], the standard DE's binomial crossover has been used for the binary representation, while, in [7], the two popular permutation crossovers POS and $TPII$ have been modified in order to use the parameter CR.

Finally, as in DE, the one-to-one selection procedure (lines 7–8) replaces the population individual x_i with the trial solution z_i if and only if z_i is fitter than x_i.

III. The Algebraic Framework for EAs

ADE's main operator, i.e., the algebraic differential evolution, is built on the basis of the formal algebraic framework for Evolutionary Algorithms (EAs) described in [3]. Here below, we recall it by: (i) providing a comprehensive background on useful group theory concepts (Section III-A), (ii) showing the connection between combinatorial search spaces and finitely generated groups (Section III-B), and (iii) providing the definitions of the operators $\oplus, \ominus, \odot$ at the basis of the algebraic differential mutation of Eq. (1) (Section III-C).

A. Group theory background

A set X is a group [18] if there exists a binary operation $\star : X \times X \to X$ fulfilling the following properties: associativity, existence of a unique neutral (or identity) element $\iota \in X$, and existence of a unique inverse $x^{-1} \in X$ for any $x \in X$.

X, equipped with $\star$, is a finitely generated group (from now on *fgg*) if there exists a finite generating set $G \subseteq X$ such that any $x \in X$ is the composition of generators from G, i.e., $x = g_1 \star g_2 \star \ldots \star g_l$ for some $g_1, g_2, \ldots, g_l \in G$.

Each $x \in X$ may have multiple minimal-length factorizations in terms of G. We denote by $||x||$ the weight of x, i.e., the length of a minimal factorization of x.

Minimal factorizations allow also to define a partial order on X. Given $x, y \in X$, we write $x \sqsubseteq y$ if, for each minimal factorization s_x of x, there exists a minimal factorization s_y of y such that s_x is a prefix of s_y.

Every *fgg* $(X, \star, G)$ has an associated graph, called "Cayley graph" and denoted by $\mathcal{C}$, whose vertices are the elements of X and there is an arc from $x \in X$ to $y \in X$ labeled by $g \in G$ if and only if $y = x \star g$.

For all $x \in X$, each path in $\mathcal{C}$, from the group identity ι to x, corresponds to a factorization of x. In fact, the arc labels in the path are the generators in a factorization of x. Clearly, shortest paths correspond to minimal factorizations. Moreover, given $x, y \in X$, $x \sqsubseteq y$ if and only if, in the graph $\mathcal{C}$, there exists at least one shortest path from ι to y passing by x.

More in general, for all pairs $x, y \in X$, any path from x to y in $\mathcal{C}$ has an algebraic interpretation: if the arc labels in the path are $\langle g_1, g_2, \ldots, g_l \rangle$, then $x \star (g_1 \star g_2 \star \ldots \star g_l) = y$. In other words: $\langle g_1, g_2, \ldots, g_l \rangle$ is a factorization of $x^{-1} \star y$.

Hence, it is now possible to define a distance d among the group elements as follows: for any pair $x, y \in X$, $d(x, y)$ is the length of a shortest path from x to y in $\mathcal{C}$ or, equivalently, $d(x, y) = ||x^{-1} \star y||$.

B. Algebra of combinatorial search spaces

Often, the search space of combinatorial optimization problems can be comprehensively represented by a *fgg*. In fact, the search space is actually the Cayley graph associated with the identified *fgg*.

The connection is as follows. Given the *fgg* $(X, \star, G)$, then X is the set of discrete solutions of the problem at hand, while two solutions $x, y \in X$ are neighbors if and only if there exists a generator $g \in G$ such that $y = x \star g$. Therefore, the generating set G, by means of the group operation $\star$, allows to derive the neighborhood relationships among the discrete solutions, which are typical of combinatorial search spaces.

For the sake of clarity, let see a concrete example of a combinatorial search space which can be represented by a *fgg*: the space of bit-strings.

Example (Space of bit-strings)**.** Let denote by $\mathbb{B}^n$ the set of the n-length bit-strings. It is easy to see that $\mathbb{B}^n$ is a group under the bitwise XOR operation, denoted by $\veebar$. Moreover, $\mathbb{B}^n$ is also finitely generated by the set U of the n bit-strings with exactly one 1-bit. Importantly, by denoting with $u_i \in U$ the generator bit-string whose i-th bit is 1 (and all the remaining bits are 0s), we have that $x \veebar u_i$ exactly corresponds to flipping the i-th bit of x. Therefore, the Cayley graph associated to the *fgg* $(\mathbb{B}^n, \veebar, U)$ is actually the n-dimensional binary hypercube, i.e., the most investigated search space for binary problems. Furthermore, the induced distance is the classical Hamming distance function.

Another widely studied concrete search space that can be represented by a *fgg* is the space of permutations which has been investigated in [4] and [7].

C. Algebraic operators

Section III-B establishes the possibility to use an algebraic language for defining elementary moves between neighboring solutions in combinatorial search spaces.

Here, we move one step beyond and we define the algebraic operators $\oplus, \ominus, \odot$ that, analogously to their Euclidean counterparts, allow to express composite moves in combinatorial spaces.

Given a generic *fgg* $(X, \star, G)$ and its associated search space graph $\mathcal{C}$, the key observation is the dichotomous interpretation of an element of X. Any $x \in X$ can be factorized and seen as a sequence of generators, hence x corresponds to a sequence of arc labels in several paths of $\mathcal{C}$. This observation is crucial, because the elements of X can be seen both as *points*, i.e., vertices in $\mathcal{C}$, and as *vectors*[1], i.e., sequences of generators in the shortest paths of $\mathcal{C}$.

Definition 1 (Algebraic addition $\oplus$)**.** Given $x, y \in X$, the discrete sum $x \oplus y$ is defined as the application of the *vector* y to the *point* x. By considering the group theory concepts described in Section III-A, $\oplus$ is formally defined as

$$x \oplus y := x \star y.$$

Definition 2 (Algebraic subtraction $\ominus$)**.** Given $x, y \in X$, the discrete difference $y \ominus x$ is the *vector* connecting x to y, i.e., the composition of the generators on a (shortest) path from x to y. By considering the group theory concepts described in Section III-A, $\ominus$ is formally defined as

$$y \ominus x := x^{-1} \star y.$$

The last algebraic operator required to completely define the algebraic differential mutation of Eq. (1) is the scalar multiplication $\odot$ which, in our framework, is defined as a stochastic operator (due to the non-uniqueness of shortest paths in a search space graph).

Definition 3 (Algebraic scalar multiplication $\odot$)**.** Given a scalar $F \geq 0$ and $x \in X$, the stochastic scalar multiplication $z = F \odot x$ is defined as randomly selecting a $z \in X$ such that the following conditions are satisfied:

(C1) $||z|| = \lceil F \cdot ||x|| \rceil$;
(C2) if $F \in [0, 1]$, then $z \sqsubseteq x$;
(C3) if $F \geq 1$, then $x \sqsubseteq z$.

One can draw a parallel between the properties (C1–C3) of $\odot$ and the classical scalar multiplication in the Euclidean space. The latter clearly satisfies a slight variant of (C1) where the Euclidean norm replaces the group weight and the ceiling is omitted. Besides, similarly to scaled vectors in the Euclidean space, (C2) and (C3) intuitively encode the idea that $z = F \odot x$ is the *vector* x scaled down or up, respectively.

Operatively, a factorization algorithm for the concrete group at hand is required. Such an algorithm can be completely stochastic, as done e.g. in [3] and [7], or partially informed by a fitness function as in [19]. Once a minimal factorization s_x of x is computed, $z = F \odot x$ is obtained by truncating or extending s_x when, respectively, $F \in [0, 1]$ or $F \geq 1$.

For further details, we refer the interested reader to [3].

IV. ADE FOR REAL VECTORS

In this section, we introduce ADE-RV, i.e., an instantiation of the abstract ADE scheme for the search space of real vectors. We follow this line: in Section IV-A we show that the

[1] Here, with "vector" we intend "free vector", i.e., a vector without point of application.

integer vectors of $\mathbb{Z}^n$ form a *fgg*, then Section IV-B introduces an enumerable subset of real vectors which is isomorphic to $\mathbb{Z}^n$, thus forming a *fgg* as well, and, finally, in Section IV-C we provide the complete algorithmic scheme of ADE-RV.

A. The finitely generated group of integer vectors

It is widely known that the set $\mathbb{Z}^n$ of the n-dimensional integer vectors is both an additive and multiplicative group. Here, we are interested in $\mathbb{Z}^n$ as an additive group. In fact, by considering the usual vector addition $+$, we have that:

- $+$ is clearly associative,
- the "all zeros" vector $\mathbf{0}$ is the neutral element, and
- for any $x \in \mathbb{Z}^n$, the opposite vector $-x$ is its inverse.

Moreover, $+$ is also commutative, thus $\mathbb{Z}^n$ is an Abelian group. Commutativity is not required by our framework, but it is useful to simplify calculations.

The fact that $\mathbb{Z}^n$ forms a group is enough to formalize the algebraic addition and subtraction. Given $x, y \in \mathbb{Z}^n$, by following the abstract Defs. 1 and 2, $\oplus$ and $\ominus$ are simply defined as:

$$x \oplus y := x + y, \tag{2}$$

$$y \ominus x := -x + y. \tag{3}$$

Clearly, Eq. (3) is the classic subtraction $y - x$, but we prefer to write $-x + y$ because here we are only considering the group structure of $\mathbb{Z}^n$.

In order to define the algebraic scalar multiplication $\odot$, we now need to: (i) show that $\mathbb{Z}^n$, equipped with $+$, is finitely generated, and (ii) devise a procedure for computing a random minimal factorization.

Let denote by $e_i \in \mathbb{Z}^n$ the vector whose i-th entry is 1, while all the rest are 0s, then it is easy to see that the set

$$E = \{e_i : i = 1, \ldots, n\} \cup \{-e_i : i = 1, \ldots, n\} \tag{4}$$

generates $\mathbb{Z}^n$. In fact, it is possible to obtain any integer vector in $\mathbb{Z}^n$ by summing up suitable generators from E.

Given $x \in \mathbb{Z}^n$, a random minimal factorization of x, in terms of the generators in E, can be computed by means of the procedure $\mathrm{RandIntFact}$ depicted in Alg. 2.

Algorithm 2 Stochastic factorization algorithm for $\mathbb{Z}^n$

```
1: function RandIntFact($x \in \mathbb{Z}^n$)
2:     $s \leftarrow \langle \cdot \rangle$            ▷ Factorization initialized to an empty list
3:     for $i \leftarrow 1, \ldots, n$ do
4:         if $x_i > 0$ then
5:             Insert $x_i$ copies of $e_i$ in $s$
6:         else if $x_i < 0$ then
7:             Insert $-x_i$ copies of $-e_i$ in $s$
8:     Randomly shuffle $s$   ▷ This makes the algorithm stochastic
9:     return $s$
```

Since the algebraic scalar multiplication $\odot$ is computed by truncating or extending a minimal factorization (see Def. 3 and the subsequent operative description given in Section III-C), then the generating set E and the $\mathrm{RandIntFact}$ algorithm are all that is required to build a concrete implementation of ADE for the search space of integer vectors.

In this context, it is interesting to note that the Cayley graph induced by the *fgg* $(\mathbb{Z}^n, +, E)$ is the classical n-dimensional integer lattice, shown in Fig. 1a (for the case $n = 2$ and restricted to the interval $[-2, 2]$ in any dimension). Here, the group weight is the classical ℓ_1 norm, while the induced distance function is the classical Manhattan or L_1 distance between integer vectors.

B. The finitely generated group of real vectors

With the aim of providing an instantiation of the abstract algebraic framework for the search space of real vectors, we first define the enumerable subset of real vectors $\mathbb{R}^n_h \subset \mathbb{R}^n$ and, then, we show that it is isomorphic to $\mathbb{Z}^n$, thus proving that $\mathbb{R}^n_h$ is a *fgg*.

Definition 4 (The discretized set of real vectors $\mathbb{R}^n_h$)**.** Given a positive parameter $h \in \mathbb{R}^+$, we define $\mathbb{R}^n_h$ as the set of real vectors whose entries are multiple of h. Formally,

$$\mathbb{R}^n_h = \{hv : v \in \mathbb{Z}^n\}.$$

Def. 4 clearly shows that $\mathbb{R}^n_h$ is isomorphic to $\mathbb{Z}^n$ by the isomorphism $x \mapsto x/h$. This proves that $\mathbb{R}^n_h$ is a *fgg* as well, for any choice of $h \in \mathbb{R}^+$. Note also that, when $h = 1$, we have the equivalence $\mathbb{R}^n_1 = \mathbb{Z}^n$.

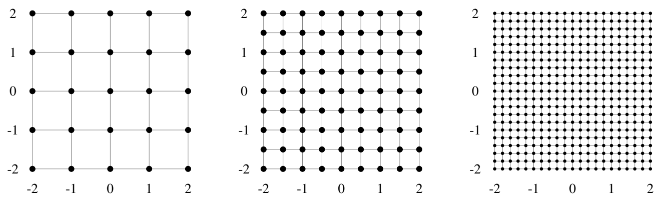
The algebraic operators $\oplus$ and $\ominus$ for $\mathbb{R}^n_h$ are defined exactly as for the integer vectors – Eqs. (2) and (3) –, while some adjustments are required for $\odot$.

First, let define the generating set E_h of $\mathbb{R}^n_h$ as follows.

$$E_h = \{he : e \in E\}, \tag{5}$$

where E is the set of integer vector generators given in Eq. (4). Therefore, we will refer to the generators in E_h as he_i and $-he_i$, for $i = 1, \ldots, n$.

The *fgg* $(\mathbb{R}^n_h, +, E_h)$ induces a search space graph with a very regular grid lattice structure that, for h approaching 0, gets more "dense" and intuitively converges to the continuous Euclidean space $\mathbb{R}^n$. Figs. 1a–1c depict, for $n = 2$, the search space graphs (restricted in any dimension to the interval $[-2, 2]$ for the sake of presentation) for $h \in \{1, 0.5, 0.2\}$.



(a) $\mathbb{Z}^2 = \mathbb{R}^2_1$ (b) $\mathbb{R}^2_{0.5}$ (c) $\mathbb{R}^2_{0.2}$

Fig. 1: Search space graphs for $\mathbb{Z}^2 = \mathbb{R}^2_1$, $\mathbb{R}^2_{0.5}$, $\mathbb{R}^2_{0.2}$

Nevertheless, it is important to recall that the algebraic scalar multiplication $\odot$ works by truncating/extending a shortest path in the search space graph. Therefore, building up a computational procedure for $\odot$ by blindly accepting any shortest path in the Cayley graph of $(\mathbb{R}^n_h, +, E_h)$ would induce a Manhattan-like distance that, in general, overestimates the

Euclidean distance. Unfortunately, it is not possible to modify the generating set E_h in order to make the induced distance function converge to the Euclidean one. Anyway, the same objective can be achieved by limiting the sampling space of the stochastic factorization algorithm. Informally, the idea is that, instead of allowing any shortest path between two points of $\mathbb{R}_h^n$, we only consider the fewer stepwise paths which try to approximate the Euclidean segment between the given points.

Therefore, instead of simply adapting Alg. 2 to work in $\mathbb{R}_h^n$, we introduce the class of *balanced factorizations* defined as in Def. 5 and computed by Alg. 3.

Definition 5 (Balanced factorizations in $\mathbb{R}_h^n$)**.** A minimal factorization s_x of $x \in \mathbb{R}_h^n$ is a *balanced factorization* if and only if, for any $e \in E_h$ appearing in s_x, the sequence of generators s_x does not contain more than one occurrence of a generator from $E_h \setminus \{e\}$ in each one of the chunks of s_x delimited by e, except (possibly) the very last chunk.

Algorithm 3 Balanced factorization algorithm for $\mathbb{R}_h^n$

```
1: function RANDREALFACT($x \in \mathbb{R}_h^n$)
2:     $s \leftarrow \langle \cdot \rangle$          ▷ Factorization initialized to an empty list
3:     $y \leftarrow x$
4:     while $y \neq \mathbf{0}$ do
5:         $R \leftarrow \{he_i \in E_h : y_i > 0\} \cup \{-he_i \in E_h : y_i < 0\}$
6:         Append to $s$ a random permutation of the set $R$
7:         $y \leftarrow y - \sum_{e \in R} e$
8:     return $s$
```

For the sake of illustration, Fig. 2 shows a limited region of the search space graph of $\mathbb{R}_{0.5}^2$.

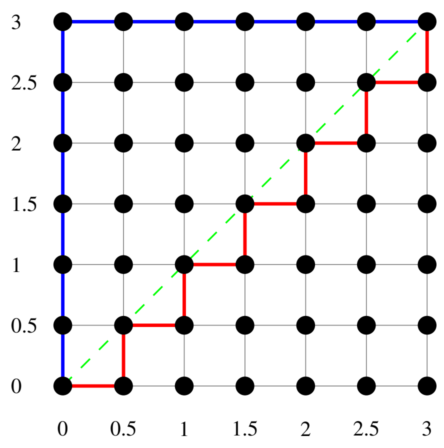


Fig. 2: Shortest paths in the search space graph of $\mathbb{R}_{0.5}^2$

Clearly, every edge in Fig. 2 is labeled by a generator from E_h.[2] The dashed green line is the Euclidean segment connecting $(0, 0)$ to $(3, 3)$. Note anyway that this green segment does not follow the graph edges, thus it does not exist as a path in our graph. Instead, both the red and blue paths are shortest paths from the identity vector $(0, 0)$ to the vector $(3, 3)$, thus they correspond to two different minimal factorizations of $(3, 3)$ in terms of E_h. However, the minimal factorization depicted in red satisfies Def. 5, thus it is a balanced factorization of $(3, 3)$, while this is not true for the minimal factorization depicted in blue. Importantly, the red path is much closer to the Euclidean segment in green than the blue path. This should now clarify why balanced factorizations allow to better approximate the Euclidean space properties. In Section V, we will anyway provide a formal proof.

Finally, though $\odot$ can be implemented by plugging Alg. 3 into the abstract operative description provided in Section III-C, for the sake of efficiency and for a quicker theoretical analysis, we provide in Alg. 4 a more direct and equivalent implementation of $\odot$ which does not require to explicitly compute the balanced factorization.

Algorithm 4 Algebraic scalar multiplication $\odot$ in $\mathbb{R}_h^n$

```
1: function ALGSCALARMULT($F > 0$, $x \in \mathbb{R}_h^n$)
2:     $z \in \mathbb{R}_h^n$ will be the result, i.e., $z = F \odot x$
3:     $w_x \leftarrow \sum_{i=1}^n \frac{|x(i)|}{h}$            ▷ $w_x$ is the group weight of $x$
4:     $w_t \leftarrow \lceil F \cdot w_x \rceil$            ▷ $w_t$ is the target weight for $z$
5:     for $i \leftarrow 1, \dots, n$ do
6:         $z(i) \leftarrow h \cdot \left\lceil F \cdot \frac{x(i)}{h} \right\rceil$
7:     $w_z \leftarrow \sum_{i=1}^n \frac{|z(i)|}{h}$     ▷ $w_z$ is the current group weight of $z$
8:     if $w_z > w_t$ then     ▷ If target weight not matched, adjust $z$
9:         $S \leftarrow w_z - w_t$ rand. sel. dimensions from $\{i : z(i) \neq 0\}$
10:        for all $i \in S$ do
11:            $z(i) \leftarrow z(i) - h \cdot \text{sign}(z(i))$
12:    return $z$                     ▷ Here, we have that $z = F \odot x$
```

Alg. 4 computes, in line 3, the group weight of x while, in line 4, it computes the target group weight for $z = F \odot x$ according to condition (C1) of Def. 3. In lines 5–6, z is initially set by evenly and proportionally considering all the dimensions. However, this setting may result in a z whose group weight is larger than the target one[3]. If this is the case, in lines 8–11, a suitable number of different dimensions – where z is not null (line 9) – is randomly selected and z is adjusted on these dimensions. Actually, line 11 corresponds to implicitly remove a generator.

The returned z is guaranteed: (i) to satisfy all the conditions provided in Def. 3, and (ii) to have a balanced factorization that is the truncation/extension of a balanced factorization of x. Furthermore, Alg. 4 can be implemented in linear time. Therefore, the algebraic differential mutation of $\mathbb{R}_h^n$ will be theoretically and experimentally analyzed by using the $\odot$ implementation of Alg. 4.

C. The ADE-RV algorithmic scheme

ADE-RV is the instantiation of the abstract ADE scheme for the *fgg* $(\mathbb{R}_h^n, +, E_h)$. Its pseudo-code is provided in Alg. 5.

ADE-RV has four parameters: the h value which regulates how dense is the search space graph (see Section IV-B),

[2]Actually, in Fig. 2: an horizontal edge corresponds to the generator he_1 ($-he_1$) when crossed from left-to-right (right-to-left), while a vertical edge corresponds to the generator he_2 ($-he_2$) when crossed from down-to-up (up-to-down).

[3]This is due to the well known property of the ceiling function, i.e., sum-of-ceilings is greater or equal than ceiling-of-sum.

Algorithm 5 Pseudo-code of ADE-RV

```
1:  function ADE-RV($h > 0$, $N \in \mathbb{N}^+$, $F \ge 0$, $CR \in [0,1]$)
2:      Randomly initialize $N$ solutions $x_1, \dots, x_N \in \mathbb{R}_h^n$
3:      while termination criterion is not satisfied do
4:          for $i \leftarrow 1, \dots, N$ do
5:              $y_i \leftarrow x_{r_0} \oplus F \odot (x_{r_1} \ominus x_{r_2})$
6:              $z_i \leftarrow$ BinomialCrossover($x_i, y_i, CR$)
7:          for $i \leftarrow 1, \dots, N$ do
8:              if $f(z_i) < f(x_i)$ then        ▷ Minimization is assumed
9:                  $x_i \leftarrow z_i$
10:     return $x_{\text{best}}$
```

the population size N, the scale factor F, and the crossover strength CR. The latter three are exactly the same parameters of the original DE scheme and play the same roles.

The population is initialized in line 2. Usually, upper and lower bounds $l, u \in \mathbb{R}$ are provided for any dimension. Hence, for each dimension $1 \le j \le n$ of any individual $1 \le i \le N$, $x_i(j)$ is randomly sampled from $[l, u]$ and then approximated to the closer multiple of the given h. This latter step makes $x_i \in \mathbb{R}_h^n$.

The algebraic differential mutation is computed as in line 5. As previously discussed, the algebraic operators $\oplus$ and $\ominus$ are the usual vector addition and subtraction, while $\odot$ is implemented as in Alg. 4. For the sake of simplicity, line 5 considers the most widely adopted mutation variant "rand/1".

After the algebraic differential mutation, any population individual x_i undergoes a crossover phase with its corresponding mutant y_i in order to produce the trial solution z_i. In line 6, the same binomial crossover of the original DE is considered. Therefore, any dimension $j = 1, \dots, n$ of z_i is set according to

$$z_i(j) \leftarrow \begin{cases} y_i(j) & \text{if } r_j < CR \text{ or } j = t, \\ x_i(j) & \text{otherwise,} \end{cases} \tag{6}$$

where: r_j is randomly generated in $[0, 1)$, t is a dimension randomly selected for each individual (to ensure that at least one dimension of the mutant is inherited by z_i), while $CR \in [0, 1]$ is the crossover strength parameter of ADE-RV.

Finally, the same one-to-one selection of the original DE is performed in lines 7–9: the fitter between x_i and z_i enters the next generation population.

V. Theoretical proof for ADE-RV operators

ADE-RV is an instantiation of the abstract ADE scheme which navigates the (giant) search space graph induced by $\mathbb{R}_h^n$. Hence, ADE-RV can be seen as a discretized variant of the original DE which works in the continuous $\mathbb{R}^n$ search space.

In Section IV-B, we have already intuitively stated that $\mathbb{R}_h^n$ converges to $\mathbb{R}^n$ for h approaching 0. Therefore, it is reasonable to expect that the search behaviour of ADE-RV converges to that of the original DE scheme. In this section, we move beyond the intuitions and we theoretically prove that the genetic operators of ADE-RV converge to those of the continuous DE scheme by also assessing their approximation errors.

ADE-RV, as introduced in Section IV-C, is composed of four procedures: initialization, algebraic differential mutation, binomial crossover, and one-to-one selection. The latter two are exactly the same procedures used in DE, therefore their approximation error is null. Hence, here below we investigate the approximation errors $\varepsilon_{\text{init}}$ and ε_{mut} due to, respectively, our discretized initialization and algebraic differential mutation.

Theorem 1. *For h approaching 0, the discretized initialization of ADE-RV converges to the continuous initialization of DE.*

Proof.
The discretized initialization of ADE-RV (see Section IV-C) works in two steps: (i) it generates a vector $\hat{x} \in \mathbb{R}^n$ uniformly at random in a given interval, and then (ii) it rounds $\hat{x}$ to the closer multiple of h in any dimension, thus forming $x \in \mathbb{R}_h^n$. Formally, for any dimension $i = 1, \dots, n$, $x_i = \text{round}\left(\frac{\hat{x}_i}{h}\right) \cdot h$.

Now, we define the approximation error $\varepsilon_{\text{init}}$ as the Euclidean norm of the difference between x and $\hat{x}$, i.e.,

$$\varepsilon_{\text{init}} = ||x - \hat{x}||_2 = \sqrt{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}.$$

By construction, we have that, for every term in the sum, $x_i - \hat{x}_i \le \frac{h}{2}$. By simple calculation, we have that $\varepsilon_{\text{init}}$ is a function of both h and n bounded as follows:

$$\varepsilon_{\text{init}}(h, n) \le \frac{h}{2}\sqrt{n}.$$

Therefore, $\lim_{h \to 0^+} \varepsilon_{\text{init}}(h, n) = 0$ for any dimensionality $n \in \mathbb{N}^+$. This proves Th. 1. □

Theorem 2. *For h approaching 0, the algebraic differential mutation of ADE-RV converges to the continuous differential mutation of DE.*

Proof.
Given $x_{r_0}, x_{r_1}, x_{r_2} \in \mathbb{R}_h^n$ and $F > 0$, the algebraic differential mutation of ADE-RV computes a mutant $y \in \mathbb{R}_h^n$ as

$$y = x_{r_0} \oplus F \odot (x_{r_1} \ominus x_{r_2}) = x_{r_0} + F \odot (x_{r_1} - x_{r_2}),$$

while the continuous differential mutation of DE computes a mutant $\hat{y} \in \mathbb{R}^n$ as

$$\hat{y} = x_{r_0} + F \cdot (x_{r_1} - x_{r_2}).$$

Now, we define the approximation error ε_{mut} as the Euclidean norm of the difference between y and $\hat{y}$, i.e., $\varepsilon_{\text{mut}} = ||y - \hat{y}||_2$.

For the sake of explanation, let use the following notations: $\delta = x_{r_1} - x_{r_2}$, $z = F \odot \delta$, and $\hat{z} = F \cdot \delta$. Hence, by simple calculation, we have that

$$\varepsilon_{\text{mut}} = ||y - \hat{y}||_2 = ||z - \hat{z}||_2 = \sqrt{\sum_{i=1}^{n} (z_i - \hat{z}_i)^2}. \tag{7}$$

Without loss of generality, we restrict our analysis to positive values for z_i and $\hat{z}_i$. By looking at Alg. 4 (in particular, lines 6 and 11), we have that $z_i \leq h \cdot \lceil F \cdot \frac{\delta_i}{h} \rceil$. Therefore, for every term in the sum of Eq. (7), we have that

$$\begin{aligned} z_i - \hat{z}_i &\leq h \cdot \lceil F \cdot \tfrac{\delta_i}{h} \rceil - F \cdot \delta_i = \\ &= h \cdot \lceil F \cdot \tfrac{\delta_i}{h} \rceil - h \cdot F \cdot \tfrac{\delta_i}{h} = \\ &= h \left(\lceil F \cdot \tfrac{\delta_i}{h} \rceil - F \cdot \tfrac{\delta_i}{h} \right) < h. \end{aligned} \tag{8}$$

The last inequality in Eq. (8) is due to the well known property that $\lceil x \rceil - x < 1$ for any $x \in \mathbb{R}$.

By plugging Eq. (8) into Eq. (7), we have that $\varepsilon_{\mathrm{mut}}$ is a function of both h and n bounded as follows:

$$\varepsilon_{\mathrm{mut}}(h, n) < h\sqrt{n}.$$

Therefore, $\lim_{h \to 0^+} \varepsilon_{\mathrm{mut}}(h, n) = 0$ for any dimensionality $n \in \mathbb{N}^+$. This proves Th. 2. $\square$

VI. Equivalence between ADE-RV and DE

In the previous section we proved that the single operators of ADE-RV converge to their counterpart in the original DE. However, the approximation errors, though small, are not null, thus they can accumulate or cancel-out each other throughout the evolution. Therefore, we experimentally investigate the statistical equivalence of ADE-RV and DE behaviours during their evolution.

We have selected one benchmark function from each one of the five benchmark categories of the popular BBOB test suite [20]. Namely, the selected functions are f_1, f_6, f_{10}, f_{15} and f_{20} (their mnemonic names are provided in the header of Tab. I). Moreover, each function is investigated with three dimensionalities, i.e., $n \in \{10, 20, 40\}$, for a total of 15 benchmark problems.

Both ADE-RV and DE were executed 100 times per problem with a budget of 100 000 evaluations. The setting $N = 100$, $F = CR = 0.5$ is adopted, while the h parameter of ADE-RV is set as $h = {}^{10^{-8}}\!/\!\sqrt{n}$. In this way, according to Section V, the approximation error in a single ADE-RV operator is bounded by 10^{-8} regardless of the problem dimensionality. Moreover, to ensure a fair comparison, ADE-RV and DE executions are run using the same set of seeds for the random number generator.

The experiments were conducted and analyzed by using the IOHprofiler tool [21]. We remark once again that the aim of this experimental investigation is not to assess the best algorithm between ADE-RV and DE, but to prove that their search behaviours are empirically equivalent. With this aim, we have statistically analyzed both the convergence graphs and the residual time series.

Convergence graph analysis. First of all, we have visually verified that the convergence graphs of ADE-RV and DE match with each other. For the sake of illustration, Figs. 3a–3b show the mean and standard deviation of the fitness trajectories of, respectively, ADE-RV and DE executed on f_{20} with $n = 40$. It is easy to see that the two curves clearly overlap each other. The same behaviour is observed also for all the other benchmark problems considered. Furthermore, we have statistically validated the comparison by performing, for each problem, ten different Kolmogorov-Smirnov tests [22] by considering the objective values at ten equidistant stages of the evolution. Almost all of the 150 p-values were 1, while the smallest one was 0.434, thus widely accepting the null hypothesis of equivalence between the trajectories of ADE-RV and DE.

Residual analysis. To confirm the previous analysis, we have computed the mean time series of the difference between ADE-RV and DE best objective values at every single generation of the evolution. For the sake of illustration, Fig. 3c provides these residual time series of the 40D versions of all the benchmark functions. All the other dimensionalities show the same trend, i.e., the residual time series slightly oscillates around 0. To statistically validate this observation, in Tab. I we provide the mean of the time series and the p-value of the Augmented Dickley-Fuller (ADF) unit root test [23]. Hence, Tab. I clearly shows that: (i) all the time series have practically 0 mean, and (ii) the ADF tests reject the null hypothesis that a unit root is present in the time series, thus reasonably indicating their stationarity, as observed by visual inspection.

Finally, we have repeated all the aforementioned analyses by considering the jDE self-adaptive scheme [24] for the F and CR parameters. The same conclusions were drawn.

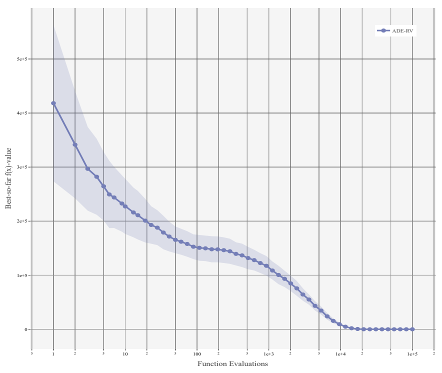
VII. Conclusion and Future Work

In this paper, we proved – both theoretically and experimentally – that the Algebraic Differential Evolution (ADE) scheme generalizes and subsumes the original Differential Evolution (DE) which, during the years, has been widely and successfully applied to a variety of problems. Therefore, the abstract ADE scheme – together with its concrete instantiations – can now be considered a full-fledged generalization of DE which allows to consistently apply the DE search behaviour across search spaces of different nature.

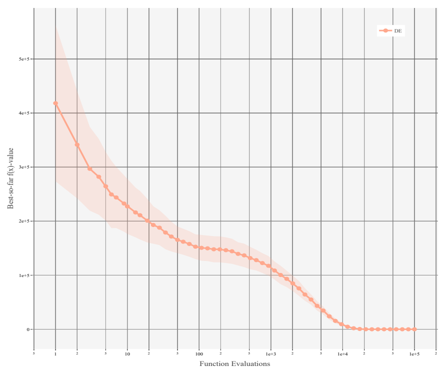
Other contributions of this work are the instantiations of the algebraic framework (for evolutionary algorithms) for the search spaces of integer vectors and real vectors. To these, must be added the previously proposed instantiations for the spaces of permutations and bit-strings [3].

This paves the way for an important future line of research. In fact, a product of multiple finitely generated groups is itself a finitely generated group [18]. This observation makes it possible to design ADE instantiations for hybrid search spaces formed by mixed discrete and continuous components, thus extending the field of application of the DE search behaviour to an even wider range of practical problems. Let think, for instance, to the tuning of machine learning hyperparameters where it is required to choose, simultaneously: boolean, integer and real-valued parameters, as well as the order of application of several computational layers.

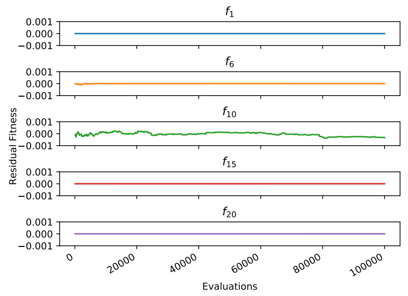
Finally, a further interesting line of research is to design a continuous DE scheme taking inspiration from the simplified ADE-RV variant which accepts any minimal factorization regardless if it is balanced or not. Concretely, this corresponds to a continuous DE that, instead of moving along the Euclidean



(a) Conv. graph for ADE-RV on f_{20}, 40D



(b) Conv. graph for DE on f_{20}, 40D



(c) Residual time series on 40D

Fig. 3: Convergence graphs and residual time series plots

TABLE I: Mean of the residual time series and p-values of the Augmented Dickley-Fuller test

	f_1 – Sphere Function			f_6 – Attractive Sector Function			f_{10} – Ellipsoidal Function			f_{15} – Rastrigin Function			f_{20} – Schwefel Function		
	10D	20D	40D	10D	20D	40D	10D	20D	40D	10D	20D	40D	10D	20D	40D
Mean	$-2 \cdot 10^{-8}$	$-5 \cdot 10^{-8}$	$-2 \cdot 10^{-5}$	$-9 \cdot 10^{-6}$	$3 \cdot 10^{-4}$	$-2 \cdot 10^{-6}$	$-5 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$-4 \cdot 10^{-7}$	$-7 \cdot 10^{-7}$	$-4 \cdot 10^{-7}$	$4 \cdot 10^{-9}$	$-7 \cdot 10^{-8}$	$-3 \cdot 10^{-7}$
p_{ADF}	10^{-7}	0.001	0.009	$4 \cdot 10^{-8}$	10^{-4}	$7 \cdot 10^{-19}$	0.019	10^{-4}	0.037	$8 \cdot 10^{-4}$	0.004	0.015	$7 \cdot 10^{-6}$	$4 \cdot 10^{-11}$	$5 \cdot 10^{-10}$

segments given by the differences of the type $x_{r_1} - x_{r_2}$, is allowed to take more diverse moves in the hypercubes whose extreme vertices are the population vectors $x_{r_!}$ and x_{r_2}.

Acknowledgment

This work was partially supported by the research grants: (i) “Università per Stranieri di Perugia – Progetto di ricerca *Artificial Intelligence for Education, Social and Human Sciences*”, and (ii) “Università per Stranieri di Perugia – *Finanziamento per Progetti di Ricerca di Ateneo — PRA 2020*”.

References

[1] V. Santucci, M. Baioletti, and A. Milani, “A differential evolution algorithm for the permutation flowshop scheduling problem with total flow time criterion,” in *Proc. of 13th Int. Conf. on Parallel Problem Solving from Nature – PPSN XIII*. Springer, 2014, pp. 161–170.

[2] R. Storn and K. Price, “Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[3] V. Santucci, M. Baioletti, and A. Milani, “An algebraic framework for swarm and evolutionary algorithms in combinatorial optimization,” *Swarm and Evolutionary Computation*, vol. 55, p. 100673, 2020.

[4] ——, “Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 682–694, 2016.

[5] ——, “Solving permutation flowshop scheduling problems with a discrete differential evolution algorithm,” *AI Communications*, vol. 29, no. 2, pp. 269–286, 2016.

[6] M. Baioletti, A. Milani, and V. Santucci, “Linear ordering optimization with a combinatorial differential evolution,” in *Proc. of 2015 IEEE Int. Conf. on Systems, Man, and Cybernetics*, 2015, pp. 2135–2140.

[7] ——, “Variable neighborhood algebraic differential evolution: An application to the linear ordering problem with cumulative costs,” *Information Sciences*, vol. 507, pp. 37–52, 2020.

[8] M. Baioletti, A. Milani, V. Santucci, and U. Bartoccini, “An experimental comparison of algebraic differential evolution using different generating sets,” in *Proc. of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO ’19, 2019, p. 1527–1534.

[9] G. Di Bari, M. Baioletti, and V. Santucci, “An experimental evaluation of the algebraic differential evolution algorithm on the single row facility layout problem,” in *Proc. of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020, p. 1678–1684.

[10] V. Santucci, M. Baioletti, G. Di Bari, and A. Milani, “A binary algebraic differential evolution for the multidimensional two-way number partitioning problem,” in *Proc. of 2019 Eur. Conf. on Evolutionary Comp. in Combinat. Optim.*, 2019, pp. 17–32.

[11] M. Baioletti, A. Milani, and V. Santucci, “Learning bayesian networks with algebraic differential evolution,” in *Proc. of 15th Int. Conf. on Parallel Problem Solving from Nature – PPSN XV*, 2018, pp. 436–448.

[12] ——, “An algebraic approach for the search space of permutations with repetition,” in *Evolutionary Computation in Combinatorial Optimization*. Cham: Springer International Publishing, 2020, pp. 18–34.

[13] L. Wang, X. Fu, Y. Mao, M. I. Menhas, and M. Fei, “A novel modified binary differential evolution algorithm and its applications,” *Neurocomputing*, vol. 98, pp. 55–75, 2012.

[14] G. Pampara, A. P. Engelbrecht, and N. Franken, “Binary Differential Evolution,” in *Proc. of 2006 IEEE International Conference on Evolutionary Computation (CEC 2006)*, 2006, pp. 1873–1879.

[15] X. Li and M. Yin, “An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure,” *Advances in Engineering Software*, vol. 55, pp. 10–31, 2013.

[16] G. Pavai and T. V. Geetha, “A survey on crossover operators,” *ACM Computing Surveys*, vol. 49, no. 4, pp. 72:1–72:43, Dec. 2016.

[17] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, “Genetic algorithms for the travelling salesman problem: A review of representations and operators,” *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[18] S. Lang, *Algebra*. Springer, 2002, vol. 211.

[19] M. Baioletti, G. Di Bari, A. Milani, and V. Santucci, “An experimental comparison of algebraic crossover operators for permutation problems,” *Fundamenta Informaticae*, vol. 174, no. 3-4, pp. 201–228, 2020.

[20] S. Finck, N. Hansen, R. Ros, and A. Auger, “Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions,” Tech. Rep., 2020. [Online]. Available: https://coco.gforge.inria.fr/downloads/download16.00/bbobdocfunctions.pdf

[21] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck, “Iohprofiler: A benchmarking and profiling tool for iterative optimization heuristics,” *arXiv e-prints:1810.05281*, Oct. 2018. [Online]. Available: https://arxiv.org/abs/1810.05281

[22] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric statistical methods*. John Wiley & Sons, 2013, vol. 751.

[23] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.

[24] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” *IEEE Trans. on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.