Proc. of Inter. Conf. on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA 2025), 7-9 August 2025, Antalya-T¨urkiye

Optimizing Evolutionary Algorithms with a Green Design Framework

Iztok Fister
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia

Iztok Fister Jr.
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia

Domen Verber
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia

Andres Iglesias
Department of Applied Mathematics and Computational Sciences
University of Cantabria
Santander, Spain

Akemi Galvez
Department of Applied Mathematics and Computational Sciences
University of Cantabria
Santander, Spain

Vili Podgorelec
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia

Damijan Novak
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia

***Abstract*—Green artificial intelligence represents a new paradigm closely aligned with the green transition in the economy. The main aim is to minimize the computational processes' carbon footprint while indirectly reducing these methods' time complexity. This study explores approaches to reducing the complexity of evolutionary algorithms, mainly when deployed on resource-constrained hardware. Green evolution strategies are introduced and evaluated against compact differential evolution using the IEEE CEC'14 (Congress on Evolutionary Computation 2014) benchmark suite in this context. The results indicate a substantial reduction in solution quality; however, this trade-off is offset by a significant decrease in time complexity (up to 25 % less time complexity). These findings highlight the potential of green evolutionary algorithms and motivate further research.**

***Keywords*—differential evolution, evolutionary algorithms, GreenAI, tinyML, optimization**

I. Introduction

Modern Artificial Intelligence (AI) tools are making massive progress in several application areas, such as object recognition [1], the discovery of new drugs [2], self-driving cars [3], robotics [4], and so on. However, this revolution also comes at a price. Beyond the financial investments in hardware and software development, the price is also expressed in the carbon footprint, which is the main consequence of using electricity for running these algorithms [5], [6].

In general, Evolutionary Algorithms (EAs) are a universal tool for solving almost all classes of optimization problems, but they usually demand huge hardware requirements in terms of time and space. These requirements are challenging to fulfill on platforms with limited hardware that have become widespread nowadays, especially with the development of micro-computers (e.g., Raspberry Pi, Arduino) and Evolutionary Robotics (ER) [7].

To overcome the problems with running EAs on limited hardware, the so-called Compact Evolutionary Algorithms (cEAs) have emerged that change the static population of individuals with a statistical description in the form of a probability density vector. The probability density vector evolves simultaneously with the maturing of the evolutionary search process, and serves for the dynamic generation of individuals entering into the operation of crossover and mutation, similar to individuals from the static population. Typically, the normal probability distribution is used in this cEA, for example, by Mininno et al. [8], who replaced the real population within compact Differential Evolution (cDE), or by Mininno et al. [9], who replaced the same within a compact Genetic Algorithm (cGA). However, Tighzert et al. [10] also tested uniform

This research was funded by the Slovenian Research and Innovation Agency, Research Core Funding No. P2-0057.

979-8-3315-3562-9/25/$31.00 ©2025 IEEE

probability distributions within their Uniform cDE (UcDE) and Uniform cGA (UcGA), and showed that using this probability distribution can even improve the results obtained by Normal distribution.

This paper brings the following two significant contributions/ innovations: (1) bridging the way between a straightforward, modest evolutionary algorithm designed with a green component on the one hand and (2) tailoring this newly developed EA to run also on limited hardware devices on the other hand. Along with this motivation, our goal is not to design a new EA that could overtake the state-of-the-art algorithms dominating different competitions, but to propose an efficient version of EA that can achieve good enough results, spend less energy, as well as run efficiently on limited hardware devices, to support its use widely also on application areas such as, Smart Agriculture [11]. Smart Agriculture is a specific area, since the landscape, weather conditions, and locations of farms often represent a constraint or bottleneck of having a direct internet connection to the cloud [12], and, thus, making the knowledge discovery on the device a must. The abbreviations used in the paper are illustrated in Table I.

TABLE I.
ABBREVIATIONS IN THE PAPER.

Meaning	Abbreviation
Artificial Intelligence	AI
Compact Evolutionary Algorithms	cEA
Compact Differential Evolution	cDE
Compact Genetic Algorithms	cGA
Congress on Evolutionary Computation	CEC
Evolutionary Algorithms	EA
Evolutionary Robotics	ER
Evolution Strategy	ES
Uniform cDE	UcDE
Uniform cGA	UcGA

II. BACKGROUND INFORMATION

This section overviews the information needed to understand the subject that follows. In line with this, a short overview of the compact Evolutionary Algorithms (cEAs) is made. The section finishes with a discussion of the Evolution Strategies (ESs), on which the proposed solution is founded.

A. Compact Evolutionary Algorithms

The main feature of the cEAs is changing the real population of individuals with sampling them dynamically from the probability distributions determined by the probability vector (more precisely, the matrix of the dimension $2 \times D$) $\mathbf{PV} = [\mu_i, \sigma_i]$ for $i = 1, \ldots, D$, where μ_i denotes the mean, σ_i the Standard Deviation and D the number of vectors' elements. Typically, two of the more used probability distributions are used, as follows:

- a Normal distribution $\mathcal{N}(\mu, \sigma)$,
- a Uniform distribution $\mathcal{U}(a, b)$.

The Normal distribution is defined with the following probability density function:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)}. \tag{1}$$

The Uniform distribution is determined by the probability density function, as follows:

$$p(x) = \begin{cases} \frac{1}{b-a}, & \text{for } a \leq x \leq b, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where a determines the lower and b the upper bound of the numerical interval, from which values can be sampled. In terms of mean μ and variance σ^2, the probability density is expressed as:

$$p(x) = \begin{cases} \frac{1}{2\sigma\sqrt{3}}, & \text{for } \sigma\sqrt{3} \leq x - \mu \leq \sigma\sqrt{3}, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where Eq. (3) describes the continuous uniform distribution. As can be seen from the equation, the feasible value x can be sampled from the following interval:

$$\mu - \sigma\sqrt{3} \leq x \leq \mu + \sigma\sqrt{3}. \tag{4}$$

The probability vector $\mathbf{PV}^{(0)}$ is initialized at the beginning of the algorithm with assigning means such as $\mu^{(0)} = 0$ and Standard Deviation $\sigma^{(0)} = \lambda$, where λ is a large number (e.g. $\lambda = 10$). In each generation, the trial individual is generated based on the $\mathbf{PV}^{(t)}$ that competes with the current best solution $\mathbf{best}$ for survival in the next generation. As a result, the winner vector $\mathbf{win}$ and loser vector $\mathbf{lose}$ are declared, and the best vector $\mathbf{best}$ is updated accordingly.

The update rule for each element of the vector $\mathbf{PV}^{(t)}$ is defined regarding the following equations:

$$\mu_i^{(t)} = \mu_i^{(t)} + \frac{1}{Np}\left(win_i^{(t)} - lose_i^{(t)}\right), \tag{5}$$

and

$$\left(\sigma_i^{(t+1)}\right)^2 = \left(\sigma_i^{(t)}\right)^2 + \left(x_i^{(t)}\right)^2 - \left(x_i^{(t+1)}\right)^2 + \frac{1}{Np}\left(\left(win_i^{(t)}\right)^2 - \left(lose_i^{(t)}\right)^2\right), \tag{6}$$

where parameter Np designates the number of virtual individuals in a population.

However, if the uniform probability distribution is selected, the lower and upper bounds of the uniformly distributed element i are calculated as follows:

$$\begin{aligned} a_i^{(t+1)} &= \mu_i^{(t)} - \sqrt{3}\sigma^{(t)}, \\ b_i^{(t+1)} &= \mu_i^{(t)} + \sqrt{3}\sigma^{(t)}. \end{aligned} \tag{7}$$

Interestingly, the trial vector $\mathbf{x}$ is obtained in cDE using the 'DE/rand/1/bin' mutation strategy, defined as:

$$\mathbf{y}^{(t)} = \mathbf{x}_t^{(t)} + F(\mathbf{x}_r^{(t)} - \mathbf{x}_s^{(t)}), \tag{8}$$

where vectors $\mathbf{x}_t$, $\mathbf{x}_r$ and $\mathbf{x}_s$ are random vectors generated from the probability distribution determined by the parameter PD, and the parameter F is a scale factor (usually $F = 0.5$), such as:

$$y_i^t = \begin{cases} \mathcal{N}(\mu_i^{(t)}, \sigma_i^{(t)} & \text{if } PD = Normal, \\ \mathcal{U}(a_i^{(t)}, b_i^{(t)}) & \text{otherwise.} \end{cases} \tag{9}$$

The mutation strategy is applied according to the parameter CR (typically $CR = 0.9$), as follows:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{y}_i^{(t)}, & \text{if rand}(0,1) \leq CR, \\ \mathbf{x}_i^{(t)}, & \text{otherwise,} \end{cases} \tag{10}$$

for $i = 1, \ldots, D$.

B. Evolution strategies

Evolution Strategies (ES) were invented in the early 1960s at the Technical University of Berlin by Rechenberg [13] and Schwefel [14]. They operate on real-valued representations of individuals, and, thus, are devoted commonly for numerical optimization. The ESs are distinguished as excellent algorithms for global optimization because they are not sensitive to falling into a local optima. On the other hand, their specialty allows the self-adaptation of the mutation parameters.

In ESs, the representation of an individual consists of two parts: (1) problem variables and (2) control parameters which are put together with the problem variables into a representation of individuals representing a solution to the problem of interest. The control parameters are not always presented depending on the ES variant. The solution, in full form, is described as follows:

$$\langle x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n, \alpha_1, \ldots, \alpha_k \rangle, \tag{11}$$

where $\langle x_1, \ldots, x_n \rangle$ denotes the problem variables, $\langle \sigma_1, \ldots, \sigma_{n_\sigma} \rangle$ the step sizes of uncorrelated mutation, and $\langle \alpha_1, \ldots, \alpha_{n_\alpha} \rangle$ are the rotation angles of correlation mutation. Thus, parameter n designates a dimension of the problem, and $k = \frac{n(n-1)}{2}$ is the number of rotation angles in the so-called correlation matrix. Due to calculation complexity, the rotation angles and corresponding implementation of the correlation mutation are avoided in this study.

Mutation is the main variation operator in ES that can operate on a single individual. Therefore this kind of variation operator is appropriate for running on limited hardware, and, consequently, complying with the demands of green AI. The simplest mutation operator in ES is an uncorrelated mutation with one step size, where the individual is represented as:

$$\mathbf{x} = \langle x_1, \ldots, x_n, \sigma \rangle, \tag{12}$$

where σ denotes the step size that is a part of the individual's representation. The modification of the individual is performed according to the following equations:

$$\begin{aligned} \sigma' &= \sigma \cdot \exp^{\tau \cdot \mathrm{N}(0,1)}, \\ x_i' &= x_i + \sigma' \cdot \mathrm{N}(0,1), \end{aligned} \tag{13}$$

where $\tau \propto \frac{1}{\sqrt{n}}$ is a learning constant. Let us emphasize the importance of the order in which arithmetical operations must be performed, i.e., at first changing the Standard Deviation σ, and, after that, changing the problem variables x_i' for $i = 1, \ldots, n$. Thus, the following relation is also considered:

$$\sigma' < \epsilon_0 \Rightarrow \sigma' = \epsilon_0, \tag{14}$$

where ϵ_0 is a predefined constant preventing the Standard Deviation from falling too close to zero.

An uncorrelated mutation with n step sizes is more complex, because, on the one hand, it assigns the step size to each problem variable, and, thus, enables searching the optimum values of each program variable independently on the other. In line with this, the representation of the individual is widened as follows:

$$\mathbf{x} = \langle x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n \rangle. \tag{15}$$

The modification of the program variables is expressed using the following equations:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot \exp^{\tau' \cdot \mathrm{N}(0,1) + \tau \cdot \mathrm{N}_i(0,1)} \\ x_i' &= x_i + \sigma_i' \cdot \mathrm{N}_i(0,1), \end{aligned} \tag{16}$$

where two learning constants have arisen: a common learning constant τ, and a dimensional learning constant τ_i. Usually, the values of these constants are set as: $\tau' \propto \frac{1}{\sqrt{2 \cdot n}}$ and $\tau \propto \frac{1}{\sqrt{2 \cdot \sqrt{n}}}$. The limitation in Eq. 14 is also valid for this mutation.

In general, there are two kinds of survivor selections that are designated symbolically with the following notations: (μ, λ) and $(\mu + \lambda)$. According to the first notation, the best μ offspring from λ newly generated can survive and enter into the next generation, while, regarding the second notation, the best μ individuals are selected from the union of parents and offspring. For green AI, where the single individual population is the preferred choice, the selection scheme is denoted as $(1 + \lambda)$.

III. Green EA algorithms

The purpose of green EAs is twofold: (1) to operate efficiently on regular hardware by saving electrical power, and consequently, producing less carbon footprint, and (2) to run the complex EAs on limited hardware. In line with this, we concentrated on EAs capable of working with small or even single, populations. As a result, almost two options can be found in the existing literature, i.e., employing compact EAs or returning to the roots of EAs and trying to see what could be reached by using one member ESa. As proposed in the corresponding literature, the probability distributions also have a significant effect on the behavior of these algorithms.

Let us mention that this study treats the DE algorithms only. Here, two DEs are proposed: eXtended compact DE (XcDE) and green ES (gES). The former incorporates the characteristics of already developed cDE [9] and UcDE [10] under the same umbrella, while the latter explores features of the ESs [15] using the mentioned uncorrelated mutations. Both algorithms support sampling individuals from three probability distributions: Normal, Uniform, and Cauchy. In the remainder of the section, the Cauchy probability distribution is introduced first. Then, the pseudo-codes of the implemented XcDE and gES are discussed.

A. Cauchy probability distribution

The Cauchy probability distribution $\mathcal{C}(m, b)$ is usually used in the EA community. The continuous distribution describes

resonance behavior, and it is defined using the following probability density function:

$$p(x) = \frac{1}{\pi} \frac{b}{(x-m)^2 + b^2}, \tag{17}$$

where the parameter b denotes half of the maximum, and m is the statistical median. In the sense of μ and σ, the former designates the mean instead of the median, while the latter the Standard Deviation instead of the half of the maximum.

B. Extended compact differential evolution

Introducing the Cauchy probability distribution demands a modification of the random vector generation according to Eq. (10), as follows:

$$y_i^t = \begin{cases} \mathcal{N}(\mu_i^{(t)}, \sigma_i^{(t)}) & \text{if } PD = Normal, \\ \mathcal{U}(a_i^{(t)}, b_i^{(t)}) & \text{if } PD = Uniform, \\ \mathcal{C}(\mu_i^{(t)}, \sigma_i^{(t)}) & \text{otherwise.} \end{cases} \tag{18}$$

In summary, the pseudo-code of the algorithm is illustrated in Algorithm 1. As evident from Algorithm 1, it is controlled using six parameters: virtual population size Np, initial Standard Deviation nu, problem dimension D, probability distribution PD, and boundary vectors $\mathbf{min\_D}$ and $\mathbf{max\_D}$.

Algorithm 1 Pseudo-code of the XcDE.

Require: Np - virtual population size, σ_0 - initial stdev, D - problem dimension, PD - probability distribution, $\mathbf{min\_D}, \mathbf{min\_D}$ - boundaries
Ensure: $\mathbf{best}$ - the best individual
$\mathbf{PV}^{(0)}$ = INIT($\mu_i^{(0)} = 0, \sigma_i^{(0)} = \sigma_0$) for $\forall i \in [1, D]$)
$\langle \mathbf{a}^{(0)}, \mathbf{b}^{(0)} \rangle$ = INIT($a_i^{(0)} = max\_D_i, b_i^{(0)} = min\_D_i$) for $\forall i \in [1, D]$)
$\mathbf{best}$ = GENERATE($P^{(0)}$)
while TERMINATECONDITION **do**
  $\langle \mathbf{x}_r, \mathbf{x}_s, \mathbf{x}_t \rangle = \begin{cases} \text{GENERATE}(a^{(t)}, b^{(t)}), & \text{if } PD = Uniform, \\ \text{GENERATE}(PV^{(t)}), & \text{otherwise.} \end{cases}$
  for i=1:D **do**
    $\mathbf{x}_i = \begin{cases} \mathbf{x}_t + F(\mathbf{x}_r + \mathbf{x}_s), & \text{if } \mathcal{U}(0,1) \leq CR, \\ best_i, & \text{otherwise.} \end{cases}$
  end for
  $\langle \mathbf{win}, \mathbf{lose} \rangle$ = COMPETE($\mathbf{x}, \mathbf{best}$)
  if $f(\mathbf{x}) = f(\mathbf{win})$ **then**
    $\mathbf{best} = \mathbf{x}$
  end if
  for each $i \in [1, D]$ **do** ▷ Update $\mathbf{PV}^{(t)}$
    $\langle \mu_i^{(t+1)}, \sigma_i^{(t+1)} \rangle$ = UPDATE $\mathbf{PV}^{(t)}$ according Eqn. (5) and (6)
    if $PD = Uniform$ **then**
      $a_i^{(t+1)} = \mu_i^{(t)} - \sqrt{3}\sigma^{(t)}$
      $b_i^{(t+1)} = \mu_i^{(t)} + \sqrt{3}\sigma^{(t)}$
    end if
  end for
end while

C. Green ES

Algorithm gES supports all three mentioned probability distributions that the PD control parameter can control. The number of generated parents (also the size of the matin pool MP) is controlled by the parameter λ. The algorithm also needs the other parameters, like problem dimension D, and the upper and lower bounds of the problem variables **a** and **b**.

The main modification of the original ES is needed, due to supporting the three probability distributions. In line with this, the functions for modifying the offspring are changed as follows:

$$\sigma^{(t+1)} = \begin{cases} \sigma^{(t)} \exp^{\tau \mathcal{N}(0,1)}, & \text{if } PD = Normal, \\ \sigma^{(t)} \exp^{\tau \mathcal{C}(0,1)}, & \text{if } PD = Cauchy, \\ \sigma^{(t)} \exp^{\tau \mathcal{N}(-1,1)}, & \text{otherwise,} \end{cases} \tag{19}$$

and

$$x_i^{(t+1)} = \begin{cases} x_i^{(t)} + \sigma^{(t+1)} \cdot \mathcal{N}(0,1), & \text{if } PD = Normal, \\ x_i^{(t)} + \sigma^{(t)} \cdot \mathcal{C}(0,1), & \text{if } PD = Cauchy, \\ x_i^{(t)} + \sigma^{(t)} \cdot \mathcal{U}(-1,1), & \text{otherwise.} \end{cases} \tag{20}$$

This means that in place of sampling the step sizes from the Normal distribution, the proposed gES also enables sampling these from Uniform and Cauchy probability distribution.

The pseudo-code of the gES is presented in Algorithm 2, from which it can be seen that this is much simpler than the corresponding XcDE.

Algorithm 2 Pseudo-code of the gES.

Require: λ - number of survivals, n - step size, σ_0 - initial stdev, D - problem dimension, PD - probability distribution, $\mathbf{a}, \mathbf{b}$ - boundaries
Ensure: $\mathbf{best}$ - the best individual
$\lambda = 7$
$\mathbf{x}^{(0)}$ = INIT($x_i^{(0)} = \text{rand}(a_i, b_i), \sigma_0$) for $\forall i \in [1, n]$)
$\mathbf{best} = \mathbf{x}^{(0)}$)
while TERMINATECONDITION **do**
  $\mathbf{MP}^{(t)}$ = GENERATEMP($\mathbf{x}^{(t)}, \mathbf{a}, \mathbf{b}, PD$) for $\forall j \in [1, \lambda]$
  $\mathbf{y}^{(t)}$ = FINDTHEBEST($\mathbf{MP}$)
  if $f(\mathbf{y}^{(t)}) \leq f(\mathbf{x}^{(t)})$ **then**
    $\mathbf{best} = \mathbf{x}^{(t)} = \mathbf{y}^{(t)}$
  end if
end while

IV. EXPERIMENTS AND RESULTS

The objective of the user experience test was to demonstrate that the results of the proposed gES are comparable to the results of the XcDE in terms of the quality, and are equal or better in terms of time complexity. In line with this, three algorithms were compared in the preliminary study, as follows:

- UcDE,
- gES using uncorrelated mutation with one step size,
- gES+ using uncorrelated mutation with n step sizes.

However, each of the mentioned algorithms supports sampling individuals from three probability distributions, i.e., Normal, Cauchy, and Uniform, that are distinguished between each other with corresponding modifiers, i.e., -N, -C, and -U attached to the name of the original algorithm. For instance, the gES with Normal probability distribution is designated as gES-N during the tests.

The parameter settings of the algorithms used during the experimental work are illustrated in Table II, from which it

TABLE II.
PARAMETER SETTING

Algorithm	PD	Parameter	Abbreviation	Value
XcDE	*	Virtual population	NP	100
	*	Initial Standard Deviation	σ_0	10
gES	*	Population size	Np	1
	*	Number of step sizes	n	1
	*	Size of mating pool	λ	7
	-N	Initial Standard Deviation	σ_0	0.05
	-C		σ_0	10.0
	*		σ_0	1.0
gES+	*	Population size	Np	1
	*	Number of step sizes	n	D
	*	Size of mating pool	λ	7
	-N	Initial standard deviation	σ_0	0.05
	-C		σ_0	10.0
	*		σ_0	1.0

can be noted that the setting of parameter size of a mating pool $\lambda = 7$ corresponds to one of the necessary conditions for self-adaptation in the ES requiring a not too strong selective pressure ($\lambda/\mu = 7$) [15]. On the other hand, setting the parameter initial Standard Deviation σ_0 has a crucial effect on the optimization results. While the setting was taken from the recommendation for XcDE in [10], the proper settings for gES algorithms were found after extensive experimental work. In ESs the parameter determines the so-called evolutionary window, which is where the evolutionary search process explore the search space the most effectively.

All the algorithms solved problems from the CEC'14 (Congress on Evolutionary Computation 2014) functions benchmark suite, whose characteristics are described in the next subsection in more detail.

A. Benchmark function suite

The CEC'14 test suite consists of 30 benchmark functions that are divided into four classes [16]:

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

Unimodal functions have a single global optimum and no local optima. The unimodal functions in this suite are non-separable and rotated. The multi-modal functions are either separable or non-separable. In addition, they are also rotated and/or shifted.

B. Hardware configuration

All the runs were made on a personal computer, IBM Lenovo, using the following configurations:

1) Processor - AMD Ryzen 7-1700 3.90 GHz × 8,
2) RAM - 16 GB,
3) Operating system - Linux Mint 19 Cinnamon.

All versions of the tested algorithms were implemented within the Eclipse CDT Framework.

C. Results

The functions of dimension $D = 10$ were adopted in the preliminary study. An analysis of the results based on the Friedman nonparametric statistical tests, in which the results obtained by a particular algorithm optimizing all benchmark functions of a specific dimension, were evaluated according to five standard statistical measures: *Best, Worst, Mean, Median,* and *StDev* values. Typically, these tests are conducted to estimate the quality of the results obtained by various nature-inspired algorithms for global optimization [17]. Indeed, the Friedman non-parametric test is a two-way analysis of variances by ranks. The statistic test is calculated and converted to ranks in the first step. Then, the null hypothesis is stated, which assumes that the medians between the ranks of all algorithms are equal. Here, a high rank value means a better algorithm [18]. The second step is performed only if a null hypothesis of a Friedman test is rejected. In this case, the post-hoc tests are conducted using the calculated ranks. Let us notice that the post-hoc analysis was not performed in our study, because we needed only the proper ranking of the algorithm's performance for the next step. Here, the test was conducted using a significance level of 0.05.

The results of the Friedman non-parametric statistical tests are presented in Fig. IV-C, that are divided into two parts: the Table shows the results of the Nemenyi and Wilcoxon post-hoc tests in analytical form, and the graph displays the results of the first post-hoc test visually. Let us mention that the base method is denoted with the symbol ‡, while the statistical significance is marked with the symbol † in the Table.

As is evident from Fig. IV-C, the best results were obtained by XcDE-N (i.e., the base method). Also the XcDE-C results are not statistically significant compared with the base method, but are statistically significant compared to the results of the other gES and gES+ algorithms. Interestingly, the results of the gES overcame the results of their counterpart gES+. Actually, these were similar, when they employed the identical probability distributions. The reason for that could be that both variants used the same initial step sizes tuned on gES variants.

The results of the cDE and gES variants regarding the time complexity are depicted in Table III, which presents the time complexity of the algorithms in the tests relating to the used probability distribution in seconds.

TABLE III.
THE COMPARATIVE ANALYSIS ACCORDING TO TIME COMPLEXITY IN SECONDS.

Algorithm	Normal		Cauchy		Uniform		Summ
	Tot.	Avg.	Tot.	Avg.	Tot.	Avg.	
XcDE	719.93	24	628.38	20.95	507.39	16.91	1,855.70
gES	527.36	17.58	504.94	16.83	446.75	14.89	1,479.05
gES+	639.8	21.33	574.44	19.15	483.5	16.12	1,697.74
Summ	1,887.09	20.97	1,707.76	18.98	1,437.64	15.97	5,032.49

TABLE IV. Numerical results of statistical tests.

Algorithm	Fri.	Nemenyi		Wilcoxon	
		CD	S.	p-value	S.
XcDE-U	4.83	[4.02,5.64]		> 0.05	
XcDE-C	4.26	[3.45,5.07]		> 0.05	†
XcDE-N	3.77	[2.96,4.58]	‡	> 0.05	‡
gES-U	4.42	[3.61,5.23]	†	> 0.05	†
gES-C	4.69	[3.88,5.5]	†	> 0.05	†
gES-N	5.42	[4.61,6.23]	†	> 0.05	†
gES-U+	5.43	[4.62,6.24]	†	$\gg 0.05$	†
gES-C+	6.61	[5.80,7.42]	†	$\gg 0.05$	†
gES-N+	6.71	[5.90,7.52]	†	$\gg 0.05$	†

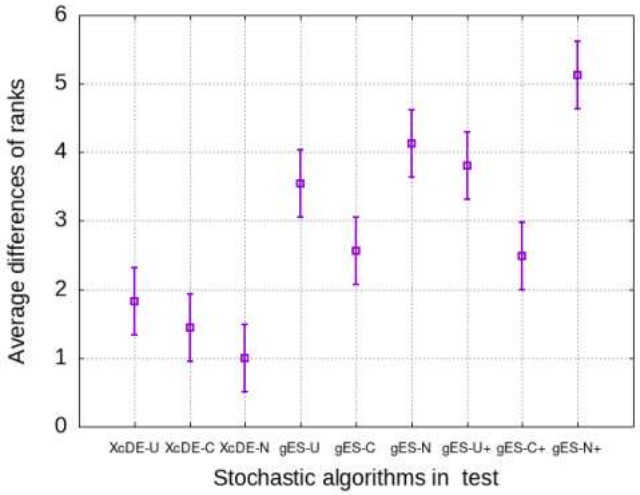

Fig. 1. The results of comparative analysis using a Nemenyi post-hoc statistical test.

As seen from the Table, the cDE algorithms are more complex w.r.t. the time consumption. The best results for these criteria were achieved by the gES algorithms which produced these even 25 % faster, while the gES+ algorithms were slightly worse (up to 10 % faster). The comparison of the probability distribution employed in these algorithms is also interesting. In this sense, it turned out that the most complex were algorithms working with Normal distribution, while running the same algorithms with Uniform distribution was more than 30 % faster, and Cauchy, which was up to 10 % faster. The main reason for the behavior lies in the fact that the calculation of the PDF for Normal distribution is complex for calculation, while the PDF for the Uniform distribution is easier.

Indeed, the time complexity was proportional to the energy consumption. As a result, we can conclude that, when we use gES, up to 25 % of energy consumption can be expected compared with the cDE. Although the obtained results are slightly worse, these are, in our opinion, good enough in practice.

V. Conclusion

In this paper, we touched on the area of green artificial intelligence, and devoted specially to the evolutionary algorithms. A new way of incorporating the green component into the evolutionary algorithm was explored, using evolution strategies to reduce the algorithm's complexity and enable the algorithm to run on limited hardware. Comprehensive experiments performed on the CEC 14 benchmark revealed that the proposed methods gES and gES+ were less complex than the original XcDE for either 25 % or 10 %, respectively, while the quality of the results are not significantly worse, and, thus, encouraged us to widen this approach to application areas. Hence, our goal for future research is to tailor this algorithm to the problem of Numerical Association Rule Mining, apply it to the application in Smart Agriculture, and in Sports domain.

Acknowledgment

A. Galvez and A. Iglesias thank the financial support from the projects PDE-GIR (Marie Sklodowska-Curie grant agreement No 778035) of the European Union's Horizon 2020 research & innovation program, and #PID2021-127073OB-I00 from the Agencia Estatal de Investigacion, Spanish Ministry of Science and Innovation (Computer Science National Program) and European Funds (MCIN/AEI/10.13039/501100011033/FEDER, EU).

References

[1] M. Taskiran, N. Kahraman, and C. E. Erdem, "Face recognition: Past, present and future (a review)," *Digital Signal Processing*, vol. 106, p. 102809, 2020.

[2] K. Huang, T. Fu, W. Gao, Y. Zhao, Y. Roohani, J. Leskovec, C. W. Coley, C. Xiao, J. Sun, and M. Zitnik, "Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development," *arXiv preprint arXiv:2102.09548*, 2021.

[3] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz *et al.*, "Self-driving cars: A survey," *Expert systems with applications*, vol. 165, p. 113816, 2021.

[4] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.

[5] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[6] I. Fister, D. Fister, and V. Podgorelec, "Profiling the carbon footprint of performance bugs," in *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*. IEEE, 2024, pp. 1–7.

[7] A. E. Eiben and J. J. E. Smith, "From evolutionary computation to the evolution of things," *Nature*, vol. 521, no. 7553, pp. 476–482, may 2015.

[8] E. Mininno, F. Neri, F. Cupertino, and D. Naso, "Compact differential evolution," *Trans. Evol. Comp*, vol. 15, no. 1, p. 32–54, feb 2011. [Online]. Available: https://doi.org/10.1109/TEVC.2010.2058120

[9] E. Mininno, F. Cupertino, and D. Naso, "Real-Valued Compact Genetic Algorithms for Embedded Microcontroller Optimization," *Trans. Evol. Comp*, vol. 12, no. 2, pp. 203–219, apr 2008. [Online]. Available: https://doi.org/10.1109/TEVC.2007.896689

[10] L. Tighzert, C. Fonlupt, O. Bruneau, and B. Mendil, "A new uniform compact evolutionary algorithms," in *2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B)*, 2017, pp. 1–6.

[11] I. Fister Jr, D. Fister, I. Fister, V. Podgorelec, and S. Salcedo-Sanz, "Time series numerical association rule mining variants in smart agriculture," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 12, pp. 16 853–16 866, 2023.
[12] A. Holzinger, I. Fister Jr, I. Fister, H.-P. Kaul, and S. Asseng, "Human-centered ai in smart farming: Towards agriculture 5.0," *IEEE Access*, 2024.
[13] I. Rechenberg, "Evolutionsstrategie," *Optimierung technischer Systeme nach Prinzipien derbiologischen Evolution*, 1973.
[14] H.-P. Schwefel, "Numerische optimierung von computer-modellen mittels der evolutionsstrategie," *(No Title)*, 1977.
[15] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Springer Publishing Company, Incorporated, 2015.
[16] J. J. Liang, B. Y. Qu, P. N. Suganthan, and Q. Chen, "Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization," Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University, Singapore, Tech. Rep., 11 2014.
[17] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, no. 1, pp. 86–92, 03 1940.
[18] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.