

Article

Binary Differential Evolution with a Limited Maximum Number of Dimension Changes

Jade Filgueira ^{1,*}, Thiago Antonini Alves ¹, Clodomir Santana ², Attilio Converti ^{3,*},
Carmelo J. A. Bastos-Filho ⁴ and Hugo Siqueira ¹

- ¹ Production Engineering Academic Department, Federal University of Technology-Parana, Ponta Grossa 84017-220, Brazil; antonini@utfpr.edu.br (T.A.A.); hugosiqueira@utfpr.edu.br (H.S.)
² Department of Internal Medicine, University of California, Davis, CA 95817, USA; clsantana@ucdavis.edu
³ Department of Civil, Chemical and Environmental Engineering, University of Genoa, Pole of Chemical Engineering, 16145 Genoa, Italy
⁴ Pernambuco Polytechnic School, University of Pernambuco, Recife 50720-001, Brazil; carmelo.filho@upe.br
* Correspondence: jade.targino@ifto.edu.br (J.F.); converti@unige.it (A.C.)

Abstract

Evolutionary Algorithms (EAs) are those based on the phenomenon of survival of the fittest. Differential Evolution (DE) is a member of this family, and it is well-suited for handling problems with real-valued variables. However, to use DE to solve binary problems, it is necessary to employ some adaptation. The primary objective of the present study is to develop a new binary version of DE. The proposed algorithm is called Binary Differential Evolution with a limited maximum number of dimension changes (NBDE), and it was tested with the OneMax Problem, five variants of the Knapsack Problem (KP), and Feature Selection (FS). The results showed that NBDE is competitive and superior to the other tested algorithms in many instances. For the 0/1 KP and 0/1 Multidimensional KP, NBDE outperforms all the other algorithms for all instances. For the FS problem, the proposed algorithm demonstrates good accuracy as its primary quality. The proposed algorithm exhibits a satisfactory performance, particularly in high-dimensional problems, which appears to be a quality inherited from the method that inspired its creation. This is particularly interesting because it provides empirical evidence that the importation of operators can perpetuate a pattern of behavior in algorithms with different structures.

Keywords: Evolutionary Algorithms; differential evolution; binary optimization problems



Academic Editor: Lorentz Jäntschi

Received: 7 September 2025

Revised: 27 September 2025

Accepted: 29 September 2025

Published: 1 October 2025

Citation: Filgueira, J.; Antonini Alves, T.; Santana, C.; Converti, A.; Bastos-Filho, C.J.A.; Siqueira, H. Binary Differential Evolution with a Limited Maximum Number of Dimension Changes. *Algorithms* **2025**, *18*, 621. <https://doi.org/10.3390/a18100621>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Evolutionary Computing (EC) is a research area within computer science that draws inspiration from the natural evolution process. To illustrate this analogy, consider an environment populated by individuals that strive for survival and reproduction. The fitness of these individuals represents their chances of survival and of multiplying. Meanwhile, in the context of a problem-solving process, we have a collection of candidate solutions, and their quality determines the likelihood that they will be retained and used as parents for constructing further candidate solution components in the next generation [1].

Evolutionary Algorithms (EAs) are those based on the phenomenon of survival of the fittest, and there are many different variants of them. Differential Evolution (DE), developed by [2], consists of mutation, crossover, and selection. The mutation operator is based on the addition of a scaled difference in two randomly selected vectors to another existing one, and the name of the method is derived from this characteristic [1].

According to [3], DE is considered one of the most effective EAs for handling problems with real-valued variables due to its simple structure, robustness, speed, and ease of use. Originally, DE was developed to operate in continuous search spaces because of the use of the scaling factor F to control the amplification of the differential variation and therefore the rate in which the population evolves, as it is a real number in the interval $[0, 2]$, but since its introduction, it has become subject of interest of many researchers, and many of them have studied forms for its binarization [3–9].

Methods of binarization using logic operators, transfer functions, or Probability Estimator Operators (PEOs) are effective; however, they add complexity that detracts from DE's original simplicity, a key characteristic for this study. Therefore, it should be preserved. That is why we chose to work with an approach that dealt with non-parametric mutation operators.

Although the current state of the research field represents great advancements in the field of EAs, particularly concerning the application of binary versions of DE, new studies and innovations in this area are still necessary given the relevance of binary combinatorial optimization problems such as the 0–1 Knapsack Problem, the uncapacitated facility location problem, the maximum coverage problem, the feature selection problem, software and hardware partitioning problems, and the Knapsack Problem with a single continuous variable [8].

In this context, the primary goal of the present study is to develop a new binary version of DE, aiming to leverage its strengths while addressing the specific requirements of binary optimization problems. It was created by substituting the original differential mutation operator for the schema proposed in [10], which is a flipping operator incorporated within a mechanism that chooses and limits the maximum number of dimensions to be flipped. Since the new method is developed by importing a mutation operator from another algorithm, the secondary goal of this study is to analyze whether this importation is capable of perpetuating the behavior observed for the donor algorithm.

The next sections of this paper are organized as follows: Section 2 brings a review of the current state of the art for DE and Binary Differential Evolution (BDE). Section 3 details the proposed algorithm. Section 4 describes in detail the methodology used in the development of the present study. Section 5 presents the results of this study, along with a discussion of the findings. Finally, Section 6 concludes this work.

2. Current State of the Research Field

DE is an evolutionary computing technique originally developed by Storn and Price in 1995 [2]. The first step of DE is the initialization of a random population of parameter vectors. After initialization, the algorithm continues searching for the optimal solution, updating the generations of individuals through the operations of Mutation, Recombination, and Selection.

At the mutation stage, for each target vector $x_{i,G}$, a mutant vector $v_{i,G+1}$ is generated according to Equation (1):

$$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) \quad (1)$$

where the indices r_1 , r_2 , and $r_3 \in \{1, 2, \dots, NP\}$ are chosen randomly and must be different from each other and also from i ; thus, the number of individuals in a population must be at least four. F is a real number belonging to the interval between 0 and 2, not including zero, which controls the amplification of the difference vector $(x_{r2,G} - x_{r3,G})$.

To increase diversity within the perturbed population, after mutation, the new vector is modified through the process of crossover, and the target vector is combined with the mutated vector using the scheme of Equation (2) to result in the trial vector:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (r(j) \leq CR) \text{ or } j = rn(i) \\ x_{ji,G} & \text{if } (r(j) > CR) \text{ or } j \neq rn(i) \end{cases} \quad (2)$$

For $j = 1, 2, \dots, D$, $r(j) \in [0, 1]$ is the j -th evaluation of a uniform random number generator. CR is the crossover rate ($CR \in [0, 1]$). $rn(i) \in (1, 2, \dots, D)$ is a randomly chosen index that ensures that $u_{i,G+1}$ receives at least one element from $v_{i,G+1}$ to ensure that new vectors are produced and the population changes.

In the selection step, a greedy selection scheme is employed. If the trial vector yields a better cost function than the target vector, it is accepted as the new parent vector for the next generation $G + 1$. Otherwise, the target vector is retained to serve as the parent for the next generation, $G + 1$.

The original DE was designed to operate in continuous search spaces; therefore, to apply it to binary problems, some modification is necessary. In the literature, various mechanisms can be found to operate this adaptation. Among them, we can nominate the use of logic operators in the substitution of arithmetic operators [4,6,9,11–16].

The Binary Differential Evolution (BDE) of Wang and Guo [12] substitutes the arithmetic operators in Equation (1) for the logic operators OR, AND, and XOR, respectively. In addition, the Scaling Factor F is a 0/1 matrix randomly chosen.

Other authors have resorted to the use of transfer functions to develop binary methods of optimization based on DE. These functions can be used to transform continuous variables into discrete ones [17], and they can be applied to the initial solutions generated [3], after mutation [18], after crossover [19], or to the factor F [20].

Another interesting way of using transfer functions to adapt DE to binary problems is presented by [21]. They developed the Angle-Modulated Differential Evolution (AMDE) using homomorphic mapping, implemented through Angle Modulation, to transfer continuous-valued spaces to binary-valued spaces.

A population of continuous individuals is generated and evolved through the original DE. Whenever a fitness evaluation is required, a bitstring is generated using Equations (3) and (4). The same procedure is used to transform the final solutions into binary vectors.

$$g(x) = \sin(2\pi(x - a) \times b \times \cos(2\pi(x - a) \times c)) + d \quad (3)$$

$$y = \begin{cases} 1 & \text{if } g(x) > 0 \\ 0 & \text{if } g(x) \leq 0 \end{cases} \quad (4)$$

where x is a single element from a set of evenly separated intervals determined by the dimension of the original binary problem, and the coefficients a , b , c , and d determine the shape of the generating function.

Engelbrecht and Pampará [22] used similar approaches to develop binary DE (binDE) and Normalization DE (NormDE). The common idea among these methods is to utilize original DE to evolve the population and generate bitstrings from the continuous individuals whenever necessary (for fitness evaluations and final solution transformation).

In binDE, concepts are borrowed from the binary Particle Swarm Optimization (PSO) of [23]. The floating-point $x_i(t)$ is used to generate a bitstring solution from Equation (5). The fitness function uses this bitstring to determine its quality. The resulting fitness is then associated with the floating-point representation of the individual.

$$y_{ij}(t) = \begin{cases} 1 & \text{if } U(0,1) < f(x_{it}(t)) \\ \text{otherwise} & \end{cases} \quad (5)$$

where $f(x_{it}(t))$ is the sigmoid function.

NormDE first normalizes the solution represented by each individual. That is, each component of each individual is linearly scaled to the range $[0, 1]$ using Equation (6).

$$x'_{ij}(t) = (x_{ij}(t) + x_i^{\min}) / (|x_i^{\min}(t)| + x_i^{\max}(t)) \quad (6)$$

where $x_i^{\min}$ and $x_i^{\max}$ are the smallest and largest component values for the i -th individual, respectively. The bitstring solution is then generated using Equation (7).

$$y_{ij}(t) = \begin{cases} 1 & \text{if } x'_{ij}(t) < 0.5 \\ \text{otherwise} & \end{cases} \quad (7)$$

Other authors have developed binary mutation operators based on Probability Estimator Operators (PEOs) [12,24–26]. Another strategy found in the literature is the development of nonparametric mutation operators, which means creating mutation operators that are not dependent on parameters such as the Scale Factor F , whose value varies linearly from 0 to 2, and therefore yields nonbinary vectors. Authors such as [7,27–29] have adopted this strategy to construct their binary versions of DE.

3. Proposed Algorithm

In the DE algorithm, the use of a continuous Scaling Factor F is responsible for generating nonbinary vectors after the mutation operation in the original DE algorithm. In fact, if a binary initial population is generated, the mutation is the only operation in DE that cannot be performed without changing the individuals into the continuous space.

So, through the analysis of the binarization methods found in the literature, that is, the use of logic operators, the use of transfer functions, the development of binary mutation operators based on the use of PEOs and the development of nonparametric mutation operators, the last alternative appears to be the simplest, as it does not imply the insertion of complex functions and operates by suppressing the one parameter responsible for disturbing binary solutions.

The authors of DE themselves consider the simplicity of the method one of its main advantages [2]; therefore, it is considered important to retain this characteristic. To achieve this goal, a literature search was conducted to identify a new non-parametric mutation operator suitable for binary spaces.

The search was not restricted to DE methods, as it is known that the combination of mechanisms belonging to different families of metaheuristics, such as in the work of [30,31], is a recurrent strategy for the improvement of algorithms' performance.

A particular mechanism was identified that not only met the expected requirements of simplicity but also demonstrated its ability to enhance the method's performance. The idea was presented by [10] and used for the development of a Novel Binary Artificial Bee colony Algorithm (NBABC).

DE suffers from loss of diversity over generations because it generates new individuals through vectorial differences between individuals in the population. As generations progress, solutions tend to align around the most promising individuals, which can lead to premature stagnation. This is one reason why it is believed that the NBABC mutation mechanism can help solve the problem of DE's loss of diversity, as individuals are mutated according to a flipping schema that does not depend on that difference.

The mechanism consists of assigning a number num_{dim} of randomly chosen dimensions from the selected food source to the current food source. This number is calculated according to Equation (8):

$$num_{dim} = \text{ceil}(\max_flips \times D) \quad (8)$$

where $\max_flips$ is a user parameter that must be set in the interval $[0, 1]$, and $\text{ceil}()$ is a function used to guarantee that at least one dimension is selected to be changed. After that, num_{dim} is randomly selected in the current food source; each one is compared to the corresponding dimensions in another randomly selected food source, and the different dimensions are set to those in the chosen food source. This procedure substitutes the equation used in the employed bees' phase and the onlooker bees' phase.

The procedure described is suitable for substituting the mutation equation in the DE algorithm, as it can introduce variability into individuals without requiring any parameters. This simple modification enables traditional DE to operate in binary spaces, resulting in a Binary Differential Evolution with a limited maximum number of dimension changes (NBDE) algorithm that is competitive or superior to many other binary algorithms and binary DE variants, as demonstrated in the Results Section. The NBDE process is described in Algorithm 1.

Algorithm 1 starts by setting the following parameters: POP_SIZE (integer), the population size; D (integer), the problem dimension; MAX_EVAL (integer), the termination budget in number of fitness evaluations; $\max_flips$ (real in $[0, 1]$), the fraction of bits allowed to change during mutation; and CR (real in $[0, 1]$), the Crossover Rate.

At each iteration, every $x_i \in \{0,1\}^D$ (the current individual at index i) is mutated to produce a vector v by copying num_{dim} positions from a uniformly sampled donor x_{r1} ($r1 \neq i$) into x_i . Next, a trial vector u is generated via binomial crossover between v and x_i . Then, greedy selection is applied between u and x_i , which means that if the fitness of u is bigger than the fitness of x_i , we set $u \rightarrow x_i$; otherwise, x_i is retained. The survivor is kept as the parent for the next generation.

Algorithm 1: NBDE Pseudocode

```

1 Set parameters
2 Initialize Population
3 for iteration in  $\text{range}(MAX\_EVAL)$ 
4   for  $i$  in  $\text{range}(POP\_SIZE)$ :
5     Randomly select one individual  $x_{r1}$  from population  $\neq i$ 
6     Create  $v$  as a copy of  $i$ 
7     Set  $\text{num}_{dim} = \text{ceil}(\max\_flips * D)$ ;
8     Select  $\text{num}_{dim}$  random dimensions  $j$  from the food source using a uniform
distribution;
9     for each selected dimension  $j$  do:
10      if  $v[j] \neq x_{r1}[j]$ 
11        $v[j] \leftarrow x_{r1}[j]$ 
12      end
13     else
14       $v[j] \leftarrow x_{r1}[j]$ 
15     end
16    end
17    Create  $u$  as the result of the crossover between  $v$  and  $i$ 
18    Apply selection between  $u$  and  $x_i$ 
19  end
20 end

```

4. Materials and Methods

The proposed NBDE was developed in Python (version 3.13.5) using the Visual Studio Code (version 1.104.2) Integrated Development Environment on a machine with Windows

10 Home Single Language (64-bit) operating system, equipped with an Intel Core i3-6006U processor, a dual-core processor with four threads, and a base frequency of 2.0 GHz. The machine has 4 GB of RAM.

NBDE was tested using benchmarks and real-world problems, as described in the topic “Selected Problems” in this section, and compared with other algorithms, as detailed in the topic “Selected Algorithms”. All the data used for the benchmark problems were generated by the authors as described in the Section “Data Generation Procedure”. The data used for the Feature Selection (FS) real-world problem were obtained from the UCI Machine Learning Repository, as detailed in the description of the FS problem.

The parameters used in the experiments are detailed in the “Parameter Configuration” Section. There are also two topics in this section that describe the methodologic procedures used for the Sensibility Analysis and the Complexity Analysis carried out, they are Topics 4.5 and 4.6, respectively.

4.1. Selected Algorithms

The selected algorithms to comparison with the NBDE include binary versions of DE like the binDE and the NormDE of [22], the AMDE of [21], the BDE of [12], and binary versions of other metaheuristics like the NBABC of [10], the Binary Particle Swarm Optimization (BPSO) of [23], the Normalized Binary Artificial Bee Colony (NormABC) of [32], and also the Genetic Algorithm (GA) of [33].

4.2. Selected Problems

To assess the performance of the proposed NBDE, we selected three types of problems: OneMax, Knapsack, and Feature Selection, as in [10]. The One Max Problem consists of maximizing the number of 1 bits in the input vector. It is useful to investigate how well algorithms designed for more complicated situations perform in benign circumstances [34].

The 0/1 Knapsack Problem (KP) consists of a set of n items, each of which has a value v_i , and some cost c_i . To solve the problem, it is necessary to select a subset of those items that maximizes the sum of the values while keeping the total cost within some given capacity C_{max} [1].

The 0/1 Multidimensional Knapsack Problem (MultiKP) is a variation of the KP. In the MultiKP, there is a set of resources with a capacity b_j , and each item consumes a given amount of each resource. The problem is to select a subset of items that maximizes the sum of the values without exceeding any of the capacities of all the resources [35]. In this study, we consider two resources.

The Multiple Knapsack Problem (MKP) is another variation of the 0/1 KP where there is more than one knapsack, each with a capacity $c_i = (i = 1, \dots, n)$ that may be different or not. The problem involves selecting disjoint subsets of items for each knapsack, ensuring that the total weight of the items in each knapsack does not exceed its capacity, while maximizing the overall profit of the selected items [36]. The selected MKP uses five knapsacks with equal capacities, and the total capacity of all the knapsacks together corresponds to the total weight of all the items.

The 0/1 Subset Sum Problem (SS) is also a variation of 0/1 KP, where we are given a set of n items with each having a weight w_j and a knapsack with a capacity c . The goal is to select a subset of items whose total weight is closest to c . The bigger the total weight, the better the quality of the subset, but never exceeding c [37]. The capacity c chosen corresponds to seventy percent of the total weight of the items, and we used a light penalty that attributes the difference between the target value and the excess weight to the fitness of any unfeasible solution.

The Multiple-Choice Knapsack Problem (MCKP) adds constraints to the KP that prohibit the inclusion of an object in the solution set if another object is selected [38]. We chose to add a constraint that determines that only one item per class can be selected and a total of 5 classes with the same number of items.

Feature Selection is a real-world problem that involves finding a subset of features within a larger set in a way that improves a predictive model, retaining original features while eliminating irrelevant or redundant ones, thereby striking a balance between dimensionality reduction and interpretability [39].

As with the benchmark problems, for the feature selection experiments, we attempted to follow the steps detailed in [10] as closely as possible. Therefore, we selected six of the UCI datasets used by the authors. The datasets are described in Table 1.

Table 1. UCI Datasets for the FS Problem.

Dataset	Instances	Features
Wine	178	13
Vehicle Silhouettes	846	18
Ionosphere	351	34
German Credit Data	1000	20
Breast Cancer Wisc.	569	30
Musk1	476	166

The K-Nearest Neighbours (KNN) algorithm with five neighbours was used as a classifier, and 10-fold cross-validation was employed to evaluate its performance on the training set. The training set is randomly selected from the data imported from UCI, corresponding to 70% of the samples for each dataset. The remaining 30% of the samples are used as the test set.

Each algorithm tries to optimize the number of selected features through 6000 fitness evaluations. At the end of each execution, the best solution found is evaluated on the test set. The fitness function used is demonstrated in Equation (9):

$$fit = \alpha \cdot accuracy + (1 - \alpha) \left(1 - \frac{\#SF}{\#TF} \right) \quad (9)$$

where accuracy is the cross-validation accuracy obtained on the training set, α is defined by the user within the interval $[0, 1]$, and it was set to 0.9. $\#SF$ is the number of selected features, and $\#TF$ is the total number of features of the dataset.

4.3. Data Generation Procedure

All the data used to carry out the experiments in this section were generated using algorithms written in Python (version 3.13.5). To ensure reproducibility, we used seeds when generating values from the random library (standard library from Python, version above). The seed selected was 42.

To obtain optimal results for comparison with those of the selected metaheuristics, the problems were solved using the Coin-or Branch and Cut (CBC) solver included in the Python wrapper OR-Tools (version 9.14.6206) developed by Google [40].

4.4. Parameter Configuration

For all the selected problems, we conducted 50 independent runs with 6000 fitness evaluations and a population of 30 individuals for standardization purposes. For the remaining parameters, we sought recommendations in the literature, except for the proposed algorithm, for which we resorted to parameterization tests.

For the BinDE, NormDE, and AMDE, we used a crossover rate $CR = 0.25$ and a Scaling Factor $F = 1$. For the AMDE, the parameters that generate the continuous individuals range from $[-1, 1]$, as indicated in [22]. For the BDE, we adopted the value indicated by its authors in [12], a $CR = 0.5$. For our proposed NBDE, a parametrization test indicated a $CR = 0.5$ and a maximum number of dimensions to be changed of $n = 0.6$, and there was no F .

For the NBAC, $n = 0.1$, and the trial limit is 50 [10]. For the NormABC, the trial limit is set to 500 iterations [32]. For the GA, we adopted a mutation rate of 0.01 with bit-flip mutation, a crossover rate (CR) of 0.7 using two-point crossover, and an elitist selection strategy. This configuration was selected based on recommendations from [41] for KPs. For the BPSO, we used coefficients of acceleration $c_1 = c_2 = 2$ [23].

4.5. Sensitivity Analysis Methodologic Procedures

A Sensitivity Analysis (SA) was also carried out for the proposed method using 10 levels of the parameter *max_flips*, and the levels range from 0.1 to 1 in uniform intervals. We performed fifty executions for each problem and each level. The Fitness value used in the analysis is the average of the best fitness of each run.

For all benchmark problems, the instance with 1000 dimensions was selected for analysis, and for the FS problem, the wine dataset was chosen. Since it was desired to analyze all curves for all the problems together, the values of fitness were normalized to the interval $[0, 1]$.

4.6. Complexity Analysis Methodologic Procedures

A complexity analysis (CA) was carried out to assess how the increase in the number of dimensions influences the execution time (scalability) of the proposed method and to compare it with the other binary DE-based selected approaches. For simplicity, only the DE-based algorithms were considered in this analysis because they share a general structure, allowing the discussion to focus on the impact of each operator, which would be harder in a wide heterogeneous section.

The OneMax problem was chosen for this analysis because it does not require extensive data generation, unlike the other benchmark problems, which would demand significant resources, especially considering the increase in the number of dimensions needed for the analysis, which spanned 10 levels from 1000 to 10,000 in regular intervals. We performed fifty executions for each considered algorithm for each level. The time value used in the analysis is the average of the execution times with the best fitness for each run.

To analyze the linearity of the curves for dimensions changes $\times$ execution time we used the scaling exponent β estimation, following the elasticity definition in power-law models [42]. The exponent β was computed according to Equation (10):

$$\beta = \frac{d \log T}{d \log d} \approx \frac{\log T(d_2) - \log T(d_1)}{\log(d_2) - \log(d_1)} \quad (10)$$

where T is the mean runtime of the best executions (seconds) at the dimension level d ; this relation was used only to obtain β , using pairs of nearby dimension levels, while all other analyses and plots were carried out on the original scale T vs. d .

5. Results and Discussion

In this section, the results of the carried-out experiments are presented. For all the selected problems, we conducted 50 independent runs with 6000 fitness evaluations. The average fitness of the best solution across the 50 simulations is reported, along with standard deviations and the p -value obtained using the Wilcoxon test at a 95% confidence level.

When the p -value indicates a significant statistical difference between two methods, the symbol “>” is used to demonstrate superiority and “<” to indicate inferiority. The symbol “–” is used to indicate that there is no significant statistical difference. For the convergence graphs, the values are also extracted from the best execution.

The use of statistical tests, such as the Wilcoxon test, is essential to validate whether the differences observed between methods are statistically significant or the result of random variation. Similar strategies were employed in [43], where the Friedman test was applied to confirm that the forecasting results obtained by different neural architectures exhibited significant differences. Likewise, in [44], statistical metrics and significance tests were also used to assess the comparative performance of bio-inspired algorithms in model calibration tasks.

5.1. OneMax Problem

The OneMax Problem with 100 dimensions was used to evaluate the convergence of the selected algorithms. Most of the algorithms were able to converge before 6000 fitness evaluations. However, experiments show that the convergence for the GA and binDE algorithms occurs at approximately 60,000 fitness evaluations. Since this number is significantly higher than those of the other analyzed methods, it was chosen to stick with 6000 fitness evaluations.

Comparing NBDE with other binary algorithms that are not based on DE, we can see from Figure 1a that NBDE and NBABC exhibit similar convergence behavior, and both converge faster than the other methods.

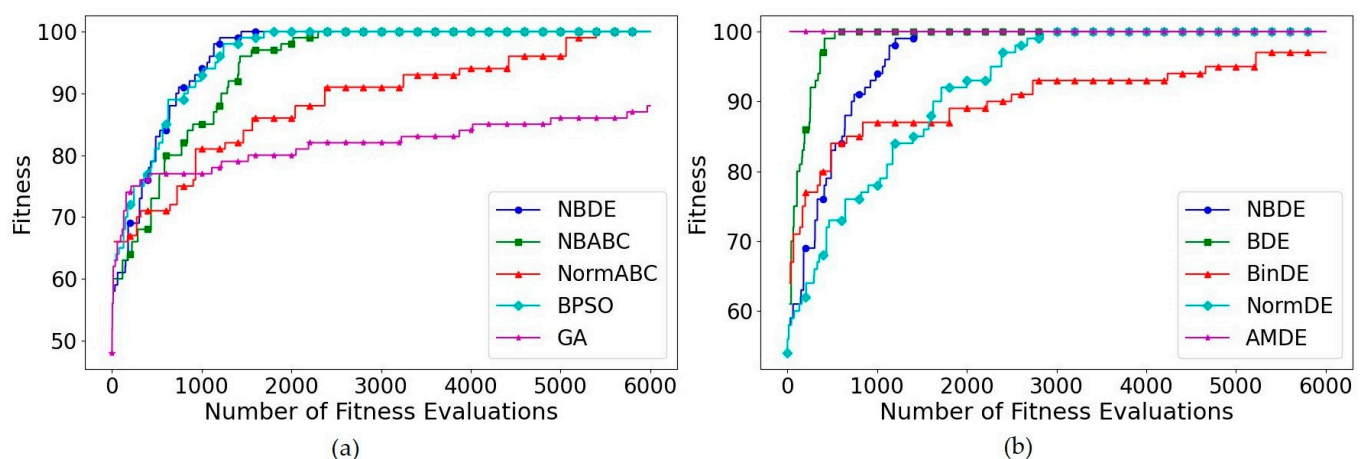


Figure 1. Convergence curves for (a) NBDE and other binary algorithms and (b) NBDE and other binary DE-based algorithms on the OneMax problem with 100 dimensions.

Figure 1b shows a comparison between NBDE and other binary DE variants, where we can see that NBDE converges faster than NormDE and binDE, and exhibits inferior behavior compared to BDE. AMDE does not present a characteristic convergence curve, probably due to the parameter setting that causes the algorithm to construct initial solutions with a large number of 1 bits, thereby reaching maximum fitness prematurely and being unable to improve afterward.

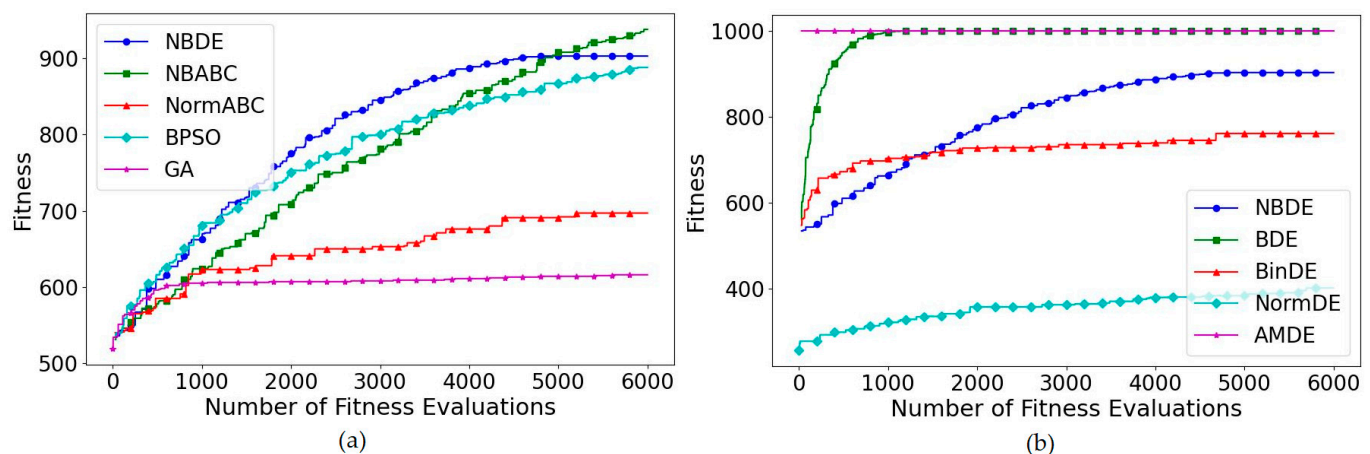
Despite the good convergence capacity of NBDE, Table 2 shows that its performance at 100 dimensions for the OneMax Problem is not one of the best of the analyzed methods, being superior only to NormABC, GA, and BinDE according to the Wilcoxon test.

Table 2. Comparison of performance between NBDE and the other selected algorithms on the OneMax problem.

Dimension		NBDE	NBABC	BPSO	NormABC	GA	binDE	NormDE	AMDE	BDE
100	Fitness	99.6 ± 0.6	100.0 ± 0.0	100.0 ± 0.0	98.4 ± 1.5	94.4 ± 2.1	94.6 ± 0.9	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	Wilcoxon	-	<	<	>	>	>	<	<	<
	Time	1.0 ± 0.0	0.6 ± 0.0	2.4 ± 0.2	0.8 ± 0.1	0.1 ± 0.0	0.5 ± 0.1	0.5 ± 0.0	0.9 ± 0.0	0.4 ± 0.0
500	Fitness	470.8 ± 5.1	497.8 ± 1.2	479.0 ± 4.1	377.4 ± 4.7	368.2 ± 7.7	396.7 ± 3.8	392.7 ± 5.4	500.0 ± 0.0	499.8 ± 0.4
	Wilcoxon	-	<	<	>	>	>	>	<	<
	Time	3.1 ± 0.1	0.9 ± 0.0	10.9 ± 0.3	0.9 ± 0.0	0.1 ± 0.0	0.9 ± 0.0	0.7 ± 0.0	1.0 ± 0.0	0.5 ± 0.0
1000	Fitness	895.4 ± 7.9	926.6 ± 6.8	873.2 ± 7.6	684.9 ± 7.1	663.8 ± 12.3	746.9 ± 4.2	694.8 ± 6.5	1000.0 ± 0.0	999.7 ± 0.6
	Wilcoxon	-	<	>	>	>	>	>	<	<
	Time	5.9 ± 0.3	1.1 ± 0.0	21.9 ± 0.2	1.0 ± 0.0	0.1 ± 0.0	1.4 ± 0.5	1.0 ± 0.0	1.1 ± 0.0	0.6 ± 0.0

For 500 dimensions, NBDE is still able to converge, whereas other algorithms, such as NBABC and BPSO, which had previously converged at 100 dimensions, apparently require a greater number of fitness evaluations. At this instance, it is possible to realize that NBDE outperforms the same methods mentioned for 100 dimensions, in addition to the NormDE method, which demonstrates a poor capacity to escalate dimensionality while maintaining good performance.

For 1000 dimensions, NBDE still demonstrates the capability to converge, indicating a good and consistent convergence capacity, even in high-dimensional problems, as shown in Figure 2a,b.

**Figure 2.** Convergence curves for (a) NBDE and other binary algorithms and (b) NBDE and other binary DE-based algorithms on the OneMax problem with 1000 dimensions.

A new analysis of the NBDE performance in Table 2 reveals that it now outperforms BPSO, as determined by the Wilcoxon test. This result demonstrates that NBDE inherits the NBABC characteristic of improving its performance in relation to other algorithms as the number of problem dimensions increases.

When we analyze the average execution time in Table 2, we can observe that NBDE has the second-highest time for 100, 500, and 1000 dimensions behind only BPSO. It does not necessarily mean that it is inefficient, but in simple problems like OneMax, the computational costs of the additional operations can become more evident due to the simplicity of the fitness function evaluation. The same behavior may not be observed for more complex problems, where the computational cost will be more closely tied to the fitness evaluation time.

5.2. Knapsack Problem

For the following problems, we chose to focus on the method's performance, as we already discussed the convergence ability through the analysis of the OneMax problem. The results for the 0/1 Knapsack problem with 100, 500, and 1000 dimensions are presented in Table 3, where it is evident that NBDE outperforms all the other tested algorithms, as confirmed by the Wilcoxon test.

Table 3. Comparison of performance between NBDE and the other selected algorithms on the 0/1 Knapsack Problem.

Dimension		NBDE	NBABC	BPSO	NormABC	GA	binDE	NormDE	AMDE	BDE
100	Fitness	11,985.4 ± 69.1	11,721.3 ± 76.4	11,823.8 ± 59.9	11,053.2 ± 115.7	10,506.4 ± 299.2	10,594.7 ± 132.8	11,577.7 ± 97.2	9393.0 ± 205.93	11,626.3 ± 153.24
	Wilcoxon	-	>	>	>	>	>	>	>	>
	Time	1.32 ± 0.2	0.74 ± 0.1	2.47 ± 0.1	0.85 ± 0.0	0.16 ± 0.0	0.5 ± 0.0	0.62 ± 0.0	0.93 ± 0.0	0.53 ± 0.1
500	Fitness	61,321.9 ± 490.9	57,798.6 ± 507.7	59,731.829 ± 520.5	51,992.9 ± 623.3	50,465.0 ± 1060.1	48,991.5 ± 534.4	53,127.1 ± 563.1	46,378.9 ± 617.1	45,769.6 ± 740.7
	Wilcoxon	-	>	>	>	>	>	>	>	>
	Time	3.2 ± 0.2	0.94 ± 0.0	11.6 ± 0.4	0.9 ± 0.0	0.2 ± 0.0	0.6 ± 0.0	0.8 ± 0.0	1.0 ± 0.0	0.8 ± 0.2
1000	Fitness	115,132.9 ± 860.4	107,356.2 ± 874.6	11,162.3 ± 1093.1	97,138.5 ± 672.9	94,388.3 ± 1597.9	90,167.0 ± 958.1	98,503.1 ± 718.7	87,811.3 ± 781.1	86,194.3 ± 1011.3
	Wilcoxon	-	>	>	>	>	>	>	>	>
	Time	5.5 ± 0.1	1.3 ± 0.0	23.4 ± 1.1	1.1 ± 0.0	0.2 ± 0.0	0.7 ± 0.1	1 ± 0.0	1.2 ± 0.0	0.8 ± 0.1

It is also possible to observe that as the number of items in the knapsack increases, the difference between NBDE and the other algorithms' performance also increases, confirming what was previously observed for the OneMax problem, even though NBDE was unable to surpass the performance of the other algorithms.

It is acceptable for one method to perform better than others in certain types of problems and not exhibit the same behavior in a different one, even when it is considered simpler, because it may require a different parameter setting. For the sake of a fair comparison, we are unable to adjust the parameters according to the type of problem.

The time analysis reveals the same behavior observed for the OneMax problem, as shown in Table 3.

5.3. Multidimensional Knapsack Problem

The results for the 0/1 Multidimensional Knapsack problem with 100, 500, and 1000 dimensions are presented in Table 4, where it is evident that NBDE outperforms all the other tested algorithms, as confirmed by the Wilcoxon test. The time analysis, as shown in Table 4, reaffirms the consistency in the behavior of the algorithms. NBDE maintains its position as the second-highest in time consumption, as discussed for the OneMax problem.

Table 4. Comparison of performance between NBDE and the other selected algorithms on the 0/1 Multidimensional Knapsack Problem.

Dimension		NBDE	NBABC	BPSO	NormABC	GA	binDE	NormDE	AMDE	BDE
100	Fitness	3854.8 ± 25.9	3773.6 ± 35.3	3820.9 ± 32.2	3527.9 ± 54.2	3302.4 ± 121.5	3336.7 ± 45.2	3680.5 ± 40.1	2996.1 ± 60.8	3479.7 ± 102.5
	Wilcoxon	-	>	>	>	>	>	>	>	>
	Time	1.4 ± 0.1	0.8 ± 0.0	2.9 ± 0.5	0.9 ± 0.0	0.2 ± 0.0	0.6 ± 0.1	0.7 ± 0.0	1.0 ± 0.0	0.5 ± 0.0
500	Fitness	19,459.6 ± 148.9	18,180.0 ± 198.2	19,021.1 ± 173.6	16,578.6 ± 196.1	15,994.1 ± 408.1	15,496.8 ± 177.05	16,883.4 ± 203.4	14,867.9 ± 146.6	14,630.0 ± 273.8
	Wilcoxon	-	>	>	>	>	>	>	>	>
	Time	3.7 ± 0.7	1.1 ± 0.1	11.8 ± 0.2	1.0 ± 0.1	0.2 ± 0.0	0.8 ± 0.0	0.9 ± 0.1	1.2 ± 0.1	0.1 ± 0.1
1000	Fitness	35,906.2 ± 293.5	33,552.0 ± 304.1	35,143.3 ± 258.2	31,318.1 ± 261.3	30,694.4 ± 460.4	29,248.4 ± 264.1	31,685.1 ± 221.9	29,145.6 ± 215.3	28,427.6 ± 348.2
	Wilcoxon	-	>	>	>	>	>	>	>	>
	Time	5.6 ± 0.2	1.4 ± 0.1	23.1 ± 0.4	1.2 ± 0.1	0.3 ± 0.0	1.0 ± 0.1	1.1 ± 0.1	1.3 ± 0.1	0.8 ± 0.1

5.4. Multiple Knapsack Problem

The results for the 0/1 Multiple Knapsack problem with 100, 500, and 1000 dimensions are described in Table 5. On this problem, NBDE was able to outperform the algorithms NormABC, GA, and BinDE for all instances and NormDE for 500 and 1000 dimensions. NBABC, BPSO AMDE, and BDE outperformed the proposed algorithm for all instances.

Table 5. Comparison of performance between NBDE and the other selected algorithms on the 0/1 Multiple Knapsack Problem.

Dimension		NBDE	NBABC	BPSO	NormABC	GA	binDE	NormDE	AMDE	BDE
100	Fitness	15,850.5 ± 65.0	15,898.6 ± 0.8	15,896.6 ± 5.4	15,484.2 ± 202.7	15,117.7 ± 243.9	15,252.8 ± 133.1	15,898.8 ± 0.4	15,873.6 ± 9.4	15,895.4 ± 7.0
	Wilcoxon	-	<	<	>	>	>	<	<	<
	Time	1.9 ± 0.1	1.5 ± 0.1	3.1 ± 0.3	1.6 ± 0.1	1.0 ± 0.1	1.3 ± 0.2	1.4 ± 0.1	2.0 ± 0.1	1.7 ± 0.2
500	Fitness	82,023.7 ± 727.2	85,299.5 ± 235.8	83,294.5 ± 506.9	66,775.4 ± 803.6	65,071.0 ± 1480.2	69,330.0 ± 592.8	69,242.4 ± 810.2	86,040.3 ± 90.8	86,195.3 ± 65.5
	Wilcoxon	-	<	<	>	>	>	>	<	<
	Time	6.8 ± 0.3	4.6 ± 0.2	14.1 ± 0.4	4.0 ± 0.3	3.8 ± 0.2	4.5 ± 0.1	4.3 ± 0.3	6.7 ± 1.0	7.0 ± 0.5
1000	Fitness	152,406.4 ± 1131.4	156,937.7 ± 987.7	150,174.0 ± 1130.4	118,066.7 ± 1266.0	114,228.8 ± 1811.5	127,659.9 ± 797.6	119,080.0 ± 847.5	168,043.9 ± 5.1	168,114.3 ± 95.5
	Wilcoxon	-	<	<	>	>	>	>	<	<
	Time	12.5 ± 0.5	7.8 ± 0.2	27.9 ± 1.1	7.1 ± 0.5	6.9 ± 0.2	8.4 ± 1.0	7.8 ± 0.6	11.8 ± 0.6	14.0 ± 0.9

5.5. Multiple Choice Knapsack Problem

For this problem, the results are presented in Table 6, which shows that there is no significant statistical difference between most of the methods. In fact, NBDE was only statistically different from GA for all instances, AMDE for 100 and 1000 dimensions, and BDE for 1000 dimensions.

Table 6. Comparison of performance between NBDE and the other selected algorithms on the 0/1 Multiple Choice Knapsack Problem.

Dimension		NBDE	NBABC	BPSO	NormABC	GA	binDE	NormDE	AMDE	BDE
100	Fitness	1351.5 ± 36.12	1352.4 ± 40.48	1362.2 ± 40.70	1356.7 ± 45.97	1028.9 ± 135.20	1351.9 ± 41.35	1348.5 ± 39.73	1336.3 ± 39.20	1354.0 ± 38.38
	Wilcoxon	-	-	-	-	>	-	-	>	-
	Time	2.1 ± 0.1	1.7 ± 0.0	3.8 ± 0.4	2.2 ± 0.3	1.3 ± 0.0	1.7 ± 0.2	2.1 ± 0.2	2.0 ± 0.1	1.7 ± 0.4
500	Fitness	1373.6 ± 24.7	1371.3 ± 23.4	1365.3 ± 24.3	1370.9 ± 20.6	1171.2 ± 83.0	1366.1 ± 17.7	1376.14 ± 21.9	1370.9 ± 23.2	1367.3 ± 20.5
	Wilcoxon	-	-	-	-	>	-	-	-	-
	Time	3.9 ± 0.2	2.0 ± 0.0	13.0 ± 0.8	2.4 ± 0.2	1.3 ± 0.1	1.9 ± 0.1	2.3 ± 0.2	2.1 ± 0.1	1.7 ± 0.0
1000	Fitness	1391.7 ± 20.0	1390.5 ± 25.3	1384.9 ± 23.8	1390.5 ± 26.9	1195.9 ± 69.9	1385.8 ± 27.5	1387.3 ± 24.5	1387.5 ± 24.7	1381.6 ± 20.9
	Wilcoxon	-	-	-	-	>	-	-	>	>
	Time	6.1 ± 0.2	2.3 ± 0.1	24.6 ± 0.5	2.3 ± 0.1	1.3 ± 0.0	2.1 ± 0.1	2.6 ± 0.2	2.2 ± 0.1	1.8 ± 0.2

The Wilcoxon test analysis suggests that the problem in the chosen configuration may have a low difficulty level, and therefore, most methods are capable of producing similar good results.

5.6. Subset Sum Problem

Table 7 contains the results for the Subset Sum Problem, and once again, NBDE presented a very good performance for 100, 500, and 1000 dimensions. The Wilcoxon test did not find enough evidence of statistical difference between NBDE and NBAC on this problem for all instances. Since the average fitness values are very close, we consider them to be competitive. Other algorithms were also able to achieve competitive performance compared to NBDE and NBAC in some instances, such as NormABC for 100 and 500 dimensions, and NormDE and BDE for 100 dimensions.

Table 7. Comparison of performance between NBDE and the other selected algorithms on the 0/1 Subset Sum Problem.

Dimension		NBDE	NBABC	BPSO	NormABC	GA	binDE	NormDE	AMDE	BDE
100	Fitness	1726.0 ± 0.0	1726.0 ± 0.0	1415.3 ± 663.1	1726.0 ± 0.0	1725.9 ± 0.4	1725.8 ± 0.4	1726.0 ± 0.0	1725.8 ± 0.5	1726.0 ± 0.0
	Wilcoxon	-	-	>	-	>	>	-	>	-
	Time	1.0 ± 0.2	0.7 ± 0.0	2.4 ± 0.1	0.8 ± 0.1	0.2 ± 0.0	0.5 ± 0.1	0.6 ± 0.0	0.8 ± 0.1	0.7 ± 0.1
500	Fitness	8408.0 ± 0.0	8408.0 ± 0.0	8407.8 ± 0.4	8407.7 ± 0.5	8407.8 ± 0.6	0 ± 0.0	8407.9 ± 0.0	8406.3 ± 2.1	0.0 ± 0.0
	Wilcoxon	-	-	>	-	>	>	>	>	>
	Time	2.8 ± 0.1	1.0 ± 0.1	16.3 ± 3.2	0.9 ± 0.1	0.2 ± 0.0	1.0 ± 0.3	0.7 ± 0.0	1.0 ± 0.1	0.8 ± 0.0
1000	Fitness	17,401.0 ± 0.2	17,401.0 ± 0.1	0.0 ± 0.0	17,023.7 ± 2432.4	7307.9 ± 8587.8	0.0 ± 0.0	17,398.7 ± 2.5	17,398.4 ± 3.1	0.0 ± 0.0
	Wilcoxon	-	-	>	>	>	>	>	>	>
	Time	2.7 ± 0.0	1.3 ± 0.0	33.1 ± 2.4	1.0 ± 0.0	0.2 ± 0.0	1.2 ± 0.2	1.4 ± 0.3	1.2 ± 0.2	0.8 ± 0.0

In this case, the increase in dimensionality improves NBDE's performance over other algorithms, even when they are comparable for 100 dimensions. Especially for some algorithms, such as binDE and BDE, which were unable to find any solutions for 500 and 1000 dimensions. The time analysis in Table 7 follows the same pattern from the previous problems.

5.7. Feature Selection Problem

Figure 3a shows the convergence curves for NBDE and the other binary methods for the wine dataset. We chose this dataset to address the convergence of the algorithms because it is the simplest in terms of the number of instances and features. We can see that all algorithms converge before 1000 fitness evaluations. NBDE is slightly slower to converge, but it achieves higher fitness values than the other methods, demonstrating that it does not exhibit premature convergence.

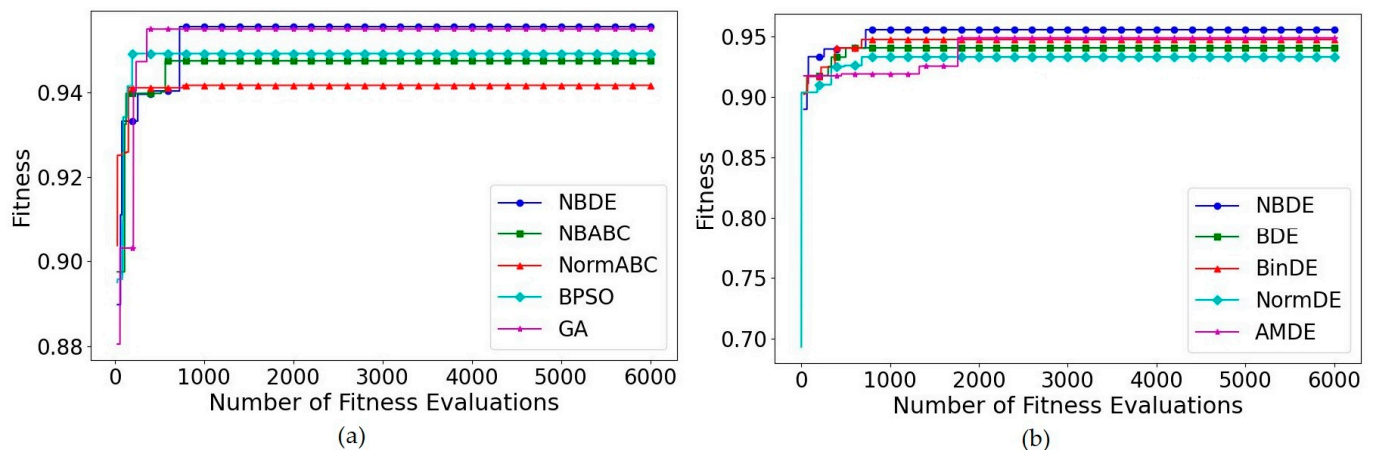
**Figure 3.** Convergence curves for (a) NBDE and other binary algorithms and (b) NBDE and other binary DE-based algorithms on the FS problem with the Wine dataset.

Figure 3b shows the convergence curves for NBDE and the other DE-based binary variants. Once again, NBDE can achieve higher fitness values than the other algorithms, although it is not the fastest method for convergence. Other algorithms, such as BDE and BinDE, converge faster but are less efficient.

Table 8 presents the results for the FS problem with the wine dataset. The Wilcoxon test did not find sufficient evidence of a statistical difference between the results obtained with NBDE and those of the other algorithms for both average fitness and accuracy. Although it is not certain, we believe that NBDE is competitive with the other algorithms, as the average fitness for all of them is very close.

Table 8. Comparison of performance between NBDE and the other selected algorithms on the FS Problem with the Wine dataset.

Dataset		NBDE	NBABC	BPSO	NormABC	GA	BinDE	NormDE	AMDE	BDE
Wine	Fitness	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0
	Wilcoxon	-	-	-	-	-	-	-	-	-
	SF	3.2 ± 0.7	3.2 ± 0.7	3.3 ± 0.7	3.4 ± 0.8	3.2 ± 0.9	3.2 ± 0.9	3.2 ± 0.7	3.2 ± 0.7	3.4 ± 0.7
	Accuracy	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	1.0 ± 0.0	0.9 ± 0.0	0.9 ± 0.0
	Wilcoxon	-	-	-	-	-	-	<	-	-
	Time	225.0 ± 4.6	226.4 ± 4.7	220.4 ± 1.4	221.0 ± 1.1	217.6 ± 1.4	229.6 ± 3.0	229.0 ± 0.8	220.1 ± 2.5	222.1 ± 5.6

Table 9 presents the same results for the vehicle silhouettes dataset. In this case, NBDE achieved good accuracy results. The Wilcoxon test at a 95% confidence level reveals that its performance is statistically superior to all other algorithms except for NormDE, which was considered superior, and BDE, for which no statistically significant difference could be observed. If we consider the average fitness, NBDE performed relatively poorly compared to the other algorithms, being superior only to AMDE and BDE. This suggests that the proposed algorithm is better at predicting the correct classes, albeit with the need for more variables to achieve this.

Table 9. Comparison of performance between NBDE and the other selected algorithms on the FS Problem with the Vehicle Silhouettes dataset.

Dataset		NBDE	NBABC	BPSO	NormABC	GA	BinDE	NormDE	AMDE	BDE
Vehicle Silhouettes	Fitness	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.01	0.7 ± 0.001	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0
	Wilcoxon	-	<	<	<	-	<	<	>	>
	SF	6.8 ± 0.6	6.7 ± 0.76	6.2 ± 0.79	6.9 ± 0.62	6.5 ± 0.7	7.0 ± 0.5	7.0 ± 0.3	6.4 ± 1.9	7.1 ± 0.7
	Accuracy	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0
	Wilcoxon	-	>	>	>	>	>	<	>	-
	Time	354.7 ± 5.6	344.1 ± 3.4	352.0 ± 56.6	350.2 ± 10.5	338.6 ± 17.3	360.1 ± 27.5	338.1 ± 7.4	337.1 ± 4.7	374.8 ± 12.3

A similar behavior was observed for the Ionosphere dataset, as shown in Table 10, where the accuracy results for NBDE were also superior to those of most other algorithms, except for AMDE, whose results do not demonstrate a statistically significant difference according to the Wilcoxon test. Meanwhile, examining the Fitness results, we can see that the proposed algorithm was only considered superior to GA BinDE, AMDE, and BDE.

Table 10. Comparison of performance between NBDE and the other selected algorithms on the FS Problem with the Ionosphere dataset.

Dataset		NBDE	NBABC	BPSO	NormABC	GA	BinDE	NormDE	AMDE	BDE
Ionosphere	Fitness	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.1	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0
	Wilcoxon	-	<	-	<	>	>	<	>	>
	SF	3.5 ± 0.7	3.7 ± 0.7	4.0 ± 1.0	3.9 ± 0.7	5.3 ± 1.6	6.8 ± 1.5	3.6 ± 0.6	3.0 ± 0.5	5.7 ± 1.8
	Accuracy	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0
	Wilcoxon	-	>	>	>	>	>	>	-	>
	Time	292.6 ± 29.0	384.6 ± 1.9	282.6 ± 6.0	287.8 ± 1.9	285.0 ± 3.5	297.1 ± 41.0	289.5 ± 15.9	256.1 ± 4.6	285.0 ± 7.1

A very different behavior was found for the German Credit Data dataset. In this case, NBDE yielded better fitness results than all the other algorithms except for NormDE, for which no statistical difference was found according to the Wilcoxon test, as shown in Table 11. For the accuracy results, there was no evidence of a statistical difference between NBDE and any of the other algorithms; however, the performances are very close, indicating that they might be competitive.

Table 11. Comparison of performance between NBDE and the other selected algorithms on the FS Problem with the German Credit Data dataset.

Dataset		NBDE	NBABC	BPSO	NormABC	GA	BinDE	NormDE	AMDE	BDE
German Credit Data	Fitness	0.8 ± 0.0	0.8 ± 0.0	0.8 ± 0.0	0.8 ± 0.0	0.8 ± 0.0	0.7 ± 0.0	0.8 ± 0.0	0.7 ± 0.0	0.7 ± 0.0
	Wilcoxon	-	>	>	>	>	>	-	>	>
	SF	14.4 ± 2.3	16.0 ± 3.03	18.7 ± 3.4	15.7 ± 2.8	19.9 ± 2.8	26.0 ± 3.0	10.8 ± 1.9	10.6 ± 3.7	31.6 ± 2.7
	Accuracy	0.7 ± 0.0	0.7 ± 0.0	0.9 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0
	Wilcoxon	-	-	-	-	-	-	-	-	-
	Time	287.5 ± 27.8	253.6 ± 12.3	263.8 ± 55.5	251.0 ± 14.6	253.8 ± 43.3	239.4 ± 3.5	389.6 ± 8.1	308.7 ± 9.6	248.2 ± 1.7

The behavior of the proposed algorithm changes once again when we consider the Breast Cancer Wisconsin (Diagnostic) dataset, as shown in Table 12. In this case, NBDE achieved superior Fitness results compared to five algorithms and superior accuracy results compared to four other algorithms, not being considered inferior to any algorithm in the analysis.

Table 12. Comparison of performance between NBDE and the other selected algorithms on the FS Problem with the Breast Cancer Wisconsin (Diagnostic) dataset.

Dataset		NBDE	NBABC	BPSO	NormABC	GA	BinDE	NormDE	AMDE	BDE
Breast Cancer Wisconsin (Diagnostic)	Fitness	1.0 ± 0.0	1.0 ± 0.0	0.9 ± 0.0	1.0 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	1.0 ± 0.0	0.9 ± 0.0	1.0 ± 0.0
	Wilcoxon	-	-	>	-	>	>	>	>	-
	SF	3.0 ± 0.0	3.0 ± 0.0	3.0 ± 0.37	3.0 ± 0.0	2.9 ± 0.62	4.3 ± 0.6	3.0 ± 0.1	3.6 ± 1.0	3.0 ± 0.1
	Accuracy	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0
	Wilcoxon	-	>	>	-	>	>	-	-	-
	Time	289.9 ± 14.2	282.4 ± 1.6	280.6 ± 2.4	291.0 ± 5.7	302.8 ± 33.2	301.2 ± 1.6	278.9 ± 6.5	274.5 ± 4.4	292.2 ± 11.5

Table 13 presents the results for the Musk 1 dataset; the behavior is more similar to that found for the Breast Cancer dataset. We observed superior fitness results to all other algorithms and superior accuracy results to five algorithms, being considered inferior only to NormDE.

Table 13. Comparison of performance between NBDE and the other selected algorithms on the FS Problem with the Musk1 dataset.

Dataset		NBDE	NBABC	BPSO	NormABC	GA	BinDE	NormDE	AMDE	BDE
Musk1	Fitness	$0.9 \pm 0.$	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.8 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.8 ± 0.0
	Wilcoxon	-	>	>	>	>	>	>	>	>
	SF	59.6 ± 5.3	67.9 ± 6.0	59.8 ± 6.6	66.5 ± 6.5	70.5 ± 5.6	88.6 ± 6.2	28.6 ± 4.7	33.3 ± 9.5	112.0 ± 6.9
	Accuracy	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.9 ± 0.0	0.89 ± 0.0	0.8 ± 0.02
	Wilcoxon	-	-	>	-	>	>	<	>	>
	Time	233.9 ± 20.2	226.7 ± 1.8	231.3 ± 31.0	221.5 ± 2.4	221.3 ± 2.3	231.0 ± 4.4	220.1 ± 1.4	215.6 ± 3.4	242.8 ± 1.8

In conclusion, we have found that NBDE has a good accuracy capacity, which means it is effective in predicting the correct classes, despite sometimes having an overall fitness score. A good accuracy capacity is preferable for algorithms when dealing with real-world applications such as FS because a good prediction ability is preferable over minimizing the number of selected features (which causes a good fitness capacity). However, NBDE yields better fitness results in typically non-noisy datasets, such as the German Credit Data, Breast Cancer, and Musk1 datasets.

The Vehicle Silhouettes and Ionosphere datasets are particularly susceptible to noise. Vehicle Silhouettes is derived from geometric vehicle contours, feature attributes obtained from digital images, which are subject to variations in segmentation, lighting, and angle, factors that introduce variability into the data. Similarly, Ionosphere, composed of radar signal return measurements, is also affected by noise because of the data acquisition process.

Regarding the execution time analysis, NBDE is always in an intermediary position; it is neither the fastest nor the slowest, meaning it does not sacrifice significant computational efficiency to achieve superior results, unlike other methods like BinDE, NormDE, and BDE, which are usually slower.

5.8. Sensitivity Analysis

A Sensitivity Analysis (SA) was conducted to evaluate how the new parameter *max_flips* influences the fitness results. The results for the SA for the proposed NBDE on all the problems are illustrated together in Figure 4, where we can see that the method is very robust for the problems FS, SS, and MCKP. In these cases, the variation in the parameter *max-flips* has little to no influence on the fitness result.

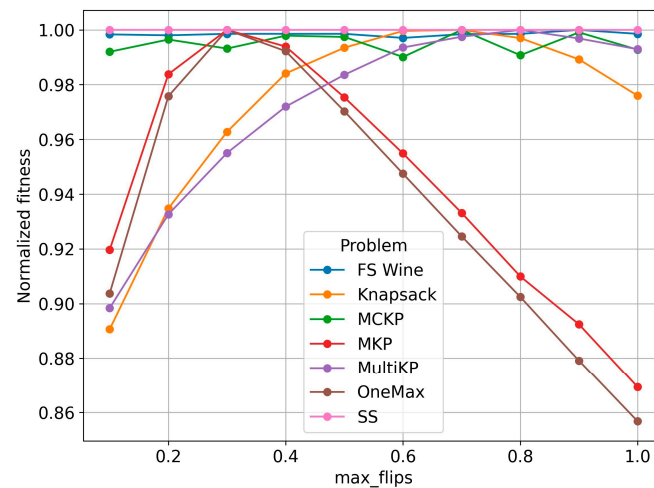


Figure 4. Sensitivity of NBDE's normalized fitness [0, 1] with respect to the parameter *max_flips*.

The algorithm is susceptible to parameter variation when applied to the problems MKP, OneMax, Knapsack, and MultiKP. The curves for the problems Knapsack and MultiKP are very similar, indicating that the increase in *max_flips* causes a fitness gain until approximately 0.6–0.7. For the OneMax and MKP, the same behavior is observed until a value of 0.3. The range between 0.4 and 0.6 appears to be a good balance, as it maintains high results for most problems without compromising those most sensitive to increased *max_flips*. This result suggests that intermediate settings offer a good compromise between robustness and performance.

5.9. Complexity Analysis

The results are analyzed considering the context of the OneMax problems with dimension changes between 1000 and 10,000. They are demonstrated in Figure 5, where we can see that NBDE, BDE, BinDE, and AMDE exhibit a behavior that appears to be linear, as shown in Table 14, indicating good scalability, as an increase in the number of dimensions appears to cause a linear increase in the average execution time.

However, when we examine the scaling exponent β , as calculated in [43], we know that the behavior of the curves is sublinear because they are smaller than 1, which means that an increase in the number of dimensions causes an increase in the average execution time that is less than linear, showing even better scalability. The same analysis can be applied to NormDE up to 8000 dimensions; beyond that point, degradation occurs, indicating worse scalability in that region.

Considering that the curves are basically parallels, we can also infer that higher curves are the ones with higher overheads (operator computational costs), and this agrees with each time analysis made at the end of Sections 5.1–5.6 for each of the benchmark problems indicating that NBDE's higher times in comparison with the other selected algorithms in these problems are mostly related to additional costs of the operators and not necessarily to the processing of the problem, and this might be attenuated for more complex problems.

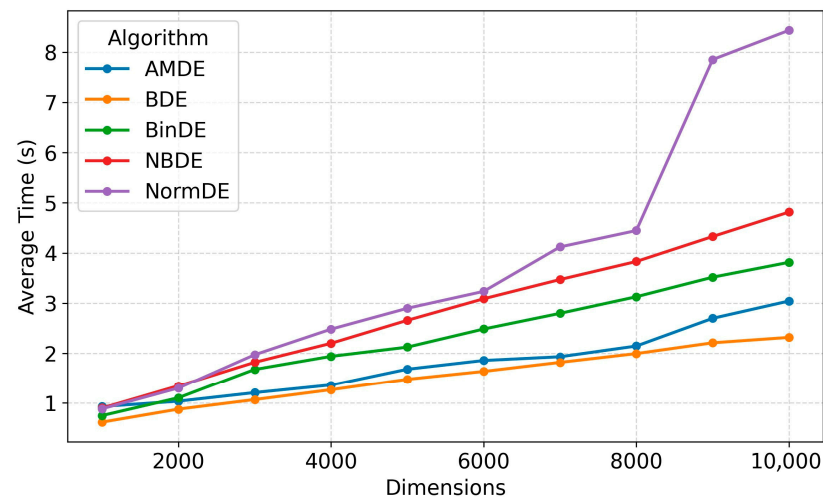


Figure 5. Complexity Analysis for NBDE and Other DE Binary DE-Based Algorithms on the One-Max Problem.

Table 14. Value of higher scaling exponent for each binary DE-based algorithm.

Algorithm	β
AMDE	0.51
BDE	0.58
BinDE	0.7
NBDE	0.73
NormDE	0.95

6. Conclusions

The primary objective of the present study was to develop a new binary version of DE that can effectively explore its strengths while addressing the specific requirements of binary optimization problems. The results on the OneMax problem demonstrate that the proposed NBDE exhibits fast convergence, even as dimensionality scales. A CA indicates good scalability also in the context of the OneMax problem. Regarding the KP family of problems, we observe that NBDE achieved a competitive or superior performance for 4 out of the five selected KP variants.

When addressing a real-world problem such as FS, NBDE also performs well in terms of accuracy, being competitive or superior to other algorithms in 93% of cases, considering all the datasets used. Regarding the same problem, the average fitness performance is competitive or superior to that of other algorithms when applied to four out of the six datasets. Apparently, NBDE does not perform well in terms of fitness when the dataset is noisy, as seen in the cases of “Vehicle Silhouettes” and “Ionosphere”.

Despite the positive results in developing a new binary method of optimization that is capable of competing or overcoming other existing ones on many problems, we believe that the main contribution of the present study resides in the presentation of empirical evidence that operators can perpetuate behavior in structurally different methods, and not only this but can also overcome the original method from which they were extracted.

We consider the produced evidence valid because it can help consolidate discussions about the identity of an algorithm, which is mainly related to its operators. Additionally, performance advances might come from modular adjustments, rather than solely from new algorithms.

The present work was born from an idea developed for another one and is living proof that a scientific study does not have to finish within itself, so for future perspectives, we see

so much potential in exploring the idea of behavior perpetuation through the substitution of the operator in question and also other operators into different algorithms to investigate how a phenomenon maintains itself in other contexts.

A second idea is to study the incorporation of an adaptive mechanism for the `max_flips` parameter to enhance the algorithm's generality and intelligence. This could be an opportunity to add further analyses such as a complexity analysis. Lastly, it is also possible to explore the application of the proposed algorithm to real-world problems, such as expansions in electricity distribution grids, as described in [45].

Author Contributions: Conceptualization, J.F. and H.S.; methodology, J.F., A.C. and H.S.; software, J.F., C.S. and H.S.; validation, A.C., C.J.A.B.-F. and H.S.; formal analysis, J.F. and T.A.A.; investigation, J.F., T.A.A. and H.S.; resources, J.F. and H.S.; data curation, J.F., T.A.A., C.S. and A.C.; writing—original draft preparation, J.F. and C.J.A.B.-F.; writing—review and editing, T.A.A., C.S. and H.S.; visualization, J.F. and T.A.A.; supervision, H.S.; project administration, H.S.; funding acquisition, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CNPq #312367/2022-8 and #311159/2023-0.

Data Availability Statement: The datasets used on the Feature Selection Problem can be found in the UCI Machine Learning Repository (<https://archive.ics.uci.edu/datasets>), accessed on 29 August 2025.

Acknowledgments: The authors gratefully acknowledge the support of UTFPR, in particular PROPPG and DIRPPG-PG.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AMDE	Angle-Modulated Differential Evolution
BinDE	Binary DE
BDE	Binary Differential Evolution
BPSO	Binary Particle Swarm Optimization
CA	Complexity Analysis
DE	Differential Evolution
EAs	Evolutionary algorithms
EC	Evolutionary Computing
FS	Feature Selection
GA	Genetic Algorithm
KP	Knapsack Problem
KNN	K-Nearest Neighbours
MultiKP	Multidimensional Knapsack Problem
MKP	Multiple Knapsack Problem
MCKP	Multiple-Choice Knapsack Problem
NBABC	Novel Binary Artificial Bee colony Algorithm
NBDE	Binary Differential Evolution with a limited maximum number of dimension changes
NormABC	Normalized Binary Artificial Bee Colony
NormDE	Normalization DE
PSO	Particle Swarm Optimization
PEOs	Probability Estimator Operators
SA	Sensitivity Analysis
SS	Subset Sum

References

1. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*, 2nd ed.; Springer: Amsterdam, The Netherlands, 2015. [\[CrossRef\]](#)
2. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [\[CrossRef\]](#)
3. Ali, I.M.; Essam, D.; Kasmarik, K. Novel binary differential evolution algorithm for knapsack problems. *Inf. Sci.* **2021**, *542*, 177–194. [\[CrossRef\]](#)
4. He, X.; Han, L. A novel binary differential evolution algorithm based on artificial immune system. In Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007. [\[CrossRef\]](#)
5. Wang, L.; Pan, Q.-K.; Suganthan, P.N.; Wang, W.-H.; Wang, Y.-M. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. *Comput. Oper. Res.* **2010**, *37*, 509–520. [\[CrossRef\]](#)
6. Fan, G.-M.; Huang, H.-J. A novel binary differential evolution algorithm for a class of fuzzy-stochastic resource allocation problems. In Proceedings of the 13th IEEE International Conference on Control & Automation, Ohrid, North Macedonia, 3–6 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 548–553. [\[CrossRef\]](#)
7. He, Z.; Xiao, L.; Huo, Z.; Wang, T.; Wang, X. Fast minimization of fixed polarity Reed-Muller expressions. *IEEE Access* **2019**, *7*, 24843–24851. [\[CrossRef\]](#)
8. He, Y.; Zhang, F.; Mirjalili, S.; Zhang, T. Novel binary differential evolution algorithm based on taper-shaped transfer functions for binary optimization problems. *Swarm Evol. Comput.* **2022**, *69*, 101022. [\[CrossRef\]](#)
9. Wang, P.; Xue, B.; Liang, J.; Zhang, M. Feature selection using diversity-based multi-objective binary differential evolution. *Inf. Sci.* **2023**, *626*, 586–606. [\[CrossRef\]](#)
10. Santana, C.J.; Macedo, M.; Siqueira, H.; Gokhale, A.; Bastos-Filho, C.J.A. A novel binary artificial bee colony algorithm. *Future Gener. Comput. Syst.* **2019**, *98*, 180–196. [\[CrossRef\]](#)
11. Wu, C.Y.; Tseng, K.Y. Engineering optimization using modified binary differential evolution algorithm. In Proceedings of the 3rd International Joint Conference on Computational Sciences and Optimization (CSO 2010), Huangshan, China, 28–31 May 2010. [\[CrossRef\]](#)
12. Wang, X.; Guo, P. A novel binary adaptive differential evolution algorithm for Bayesian network learning. In Proceedings of the 8th International Conference on Natural Computation, Chongqing, China, 29–31 May 2012. [\[CrossRef\]](#)
13. Li, H.; Zhang, L. Solving linear bilevel programming problems using a binary differential evolution. In Proceedings of the 11th International Conference on Computational Intelligence and Security, Shenzhen, China, 19–20 December 2015. [\[CrossRef\]](#)
14. Banitalebi, A.; Aziz, M.I.A.; Aziz, Z.A. A self-adaptive binary differential evolution algorithm for large scale binary optimization problems. *Inf. Sci.* **2016**, *367–368*, 487–511. [\[CrossRef\]](#)
15. Xuan, M.; Li, L.; Lin, Q.; Ming, Z.; Wei, W. A modified decomposition-based multi-objective optimization algorithm for high dimensional feature selection. In Proceedings of the 7th IEEE International Conference on Cloud Computing and Intelligence Systems, Beijing, China, 26–27 June 2021. [\[CrossRef\]](#)
16. Wei, W.; Xuan, M.; Li, L.; Lin, Q.; Ming, Z.; Coello Coello, C.A. Multiobjective optimization algorithm with dynamic operator selection for feature selection in high-dimensional classification. *Appl. Soft Comput.* **2023**, *143*, 110360. [\[CrossRef\]](#)
17. Deng, C.S.; Zhao, B.Y.; Deng, A.Y.; Liang, C.Y. Hybrid-coding binary differential evolution algorithm with application to 0–1 knapsack problems. In Proceedings of the International Conference on Computer Science and Software Engineering (CSSE), Wuhan, China, 12–14 December 2008. [\[CrossRef\]](#)
18. Xie, W.; Chi, Y.; Wang, L.; Yu, K.; Li, W. MMBDE: A two-stage hybrid feature selection method from microarray data. In Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine, Houston, TX, USA, 9–12 December 2021. [\[CrossRef\]](#)
19. He, X.; Zhang, Q.; Sun, N.; Dong, Y. Feature selection with discrete binary differential evolution. In Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI), Shanghai, China, 7–8 November 2009. [\[CrossRef\]](#)
20. Krause, J.; Lopes, H.S. A comparison of differential evolution algorithm with binary and continuous encoding for the MKP. In Proceedings of the BRICS Congress on Computational Intelligence and Brazilian Congress on Computational Intelligence (BRICS-CCI-CBIC), Ipojuca, Brazil, 8–11 September 2013. [\[CrossRef\]](#)
21. Pampará, G.; Engelbrecht, A.P.; Franken, N. Binary differential evolution. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 16–21 July 2006.
22. Engelbrecht, A.P.; Pampará, G. Binary differential evolution strategies. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Singapore, 25–28 September 2007.
23. Kennedy, J.; Eberhart, R.C. A discrete binary version of the particle swarm algorithm. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), Orlando, FL, USA, 12–15 October 1997. [\[CrossRef\]](#)
24. Parija, S.R.; Sahu, P.K.; Singh, S.S. Differential evolution for cost reduction in cellular network. In Proceedings of the International Conference on High Performance Computing and Applications (ICHPCA), Bhubaneswar, India, 22–24 December 2014. [\[CrossRef\]](#)

25. Dhaliwal, J.S.; Dhillon, J.S. A binary differential evolution based memetic algorithm to solve profit based unit commitment problem. In Proceedings of the Power India International Conference (PIICON), Kurukshetra, India, 20–22 December 2018. [CrossRef]
26. Gao, T.; Li, H.; Gong, M.; Zhang, M.; Qiao, W. Superpixel-based multiobjective change detection based on self-adaptive neighborhood-based binary differential evolution. *Expert Syst. Appl.* **2023**, *212*, 118811. [CrossRef]
27. Bidgoli, A.A.; Ebrahimpour-Komleh, H.; Rahnamayan, S. A novel multi-objective binary differential evolution algorithm for multi-label feature selection. In Proceedings of the IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019. [CrossRef]
28. Li, T.; Dong, H.; Sun, J. Binary differential evolution based on individual entropy for feature subset optimization. *IEEE Access* **2019**, *7*, 24109–24121. [CrossRef]
29. Zhang, Y.; Chen, H.; Chen, W.; Xu, L.; Li, C.; Feng, Q. Near infrared feature waveband selection for fishmeal quality assessment by frequency adaptive binary differential evolution. *Chemom. Intell. Lab. Syst.* **2021**, *217*, 104393. [CrossRef]
30. Chen, Y.; Xie, W.; Zou, X. A binary differential evolution algorithm learning from explored solutions. *Neurocomputing* **2015**, *149*, 1038–1047. [CrossRef]
31. Zhang, L.; Li, H. BPSOBDE: A binary version of hybrid heuristic algorithm for multidimensional knapsack problems. In Proceedings of the 13th International Conference on Computational Intelligence and Security, Hong Kong, China, 15–17 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 54–58. [CrossRef]
32. Pampara, G.; Engelbrecht, A.P. Binary artificial bee colony optimization. In Proceedings of the IEEE Symposium on Swarm Intelligence, Paris, France, 11–15 April 2011. [CrossRef]
33. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
34. Ackley, D.H. *A Connectionist Machine for Genetic Hillclimbing*; Kluwer Academic Publishers: Boston, MA, USA, 1987.
35. Xu, J.; Li, H.; Yin, M. Finding and exploring promising search space for the 0–1 multidimensional knapsack problem. *Appl. Soft Comput.* **2024**, *164*, 111934. [CrossRef]
36. Dell’Amico, M.; Delorme, M.; Iori, M.; Martello, S. Mathematical models and decomposition methods for the multiple knapsack problem. *Eur. J. Oper. Res.* **2019**, *274*, 886–899. [CrossRef]
37. Lai, T.-C. Worst-case analysis of greedy algorithms for the unbounded knapsack, subset-sum and partition problems. *Theor. Comput. Sci.* **1993**, *123*, 215–220. [CrossRef]
38. Zhong, T.; Young, R. Multiple choice knapsack problem: Example of planning choice in transportation. *Eval. Program Plann.* **2010**, *33*, 128–137. [CrossRef] [PubMed]
39. Liao, C.; Yang, B. A novel multi-label feature selection method based on conditional entropy and its acceleration mechanism. *Int. J. Approx. Reason.* **2025**, *185*, 109469. [CrossRef]
40. Google OR-Tools. Linear Solver Wrapper (CBC). Available online: https://developers.google.com/optimization/reference/python/linear_solver/pywraplp (accessed on 16 July 2025).
41. Rojas, I.; González, J.; Pomares, H.; Merelo, J.; Castillo, P.; Romero, G. Statistical analysis of the main parameters involved in the design of a genetic algorithm. *IEEE Trans. Syst. Man Cybern.—Part C Appl. Rev.* **2002**, *32*, 31–37. [CrossRef]
42. Weisberg, S. *Applied Linear Regression*, 4th ed.; Wiley: Hoboken, NJ, USA, 2014; ISBN 978-1-118-38608-8.
43. Giannelos, S.; Zhang, T.; Pudjianto, D.; Konstantelos, I.; Strbac, G. Investments in electricity distribution grids: Strategic versus incremental planning. *Energies* **2024**, *17*, 2724. [CrossRef]
44. Siqueira, H.; Boccato, L.; Attux, R.; Lyra Filho, C. Echo State Networks for Seasonal Streamflow Series Forecasting. In *Intelligent Data Engineering and Automated Learning—IDEAL 2012. Lecture Notes in Computer Science*; Yin, H., Costa, J.A.F., Barreto, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7435, pp. 222–229. [CrossRef]
45. Castanho, M.J.P.; Lima, D.S.; Veiga, L.M.S.; Filho, M.C.F.; Neto, E.A.B.; Silva, I.N. Performance Evaluation of Bio-Inspired Algorithms in the Calibration of Linear Models for SoC Estimation in Li-Ion Batteries. *Algorithms* **2022**, *15*, 415. [CrossRef]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.