mathematics

MDPI

Article

On the Efficacy of Ensemble of Constraint Handling Techniques in Self-Adaptive Differential Evolution

Hassan Javed [1,†], Muhammad Asif Jan [1,*,†], Nasser Tairan [2,†], Wali Khan Mashwani [1,†], Rashida Adeeb Khanum [3,†], Muhammad Sulaiman [4,†], Hidayat Ullah Khan [5,†] and Habib Shah [2,†]

1 Institute of Numerical Sciences, Kohat University of Science & Technology, Kohat 26000, Pakistan
2 College of Computer Science, King Khalid University, Abha 61321, Saudi Arabia
3 Jinnah College for Women, University of Peshawar, Peshawar 25000, Pakistan
4 Department of Mathematics, Abdul Wali Khan University, Mardan 23200, Pakistan
5 Department of Economics, Abbottabad University of Science & Technology, Abbottabad 22010, Pakistan
* Correspondence: majan.math@gmail.com or majan@kust.edu.pk
† These authors contributed equally to this work.

Received: 9 June 2019; Accepted: 10 July 2019; Published: 17 July 2019

Abstract: Self-adaptive variants of evolutionary algorithms (EAs) tune their parameters on the go by learning from the search history. Adaptive differential evolution with optional external archive (JADE) and self-adaptive differential evolution (SaDE) are two well-known self-adaptive versions of differential evolution (DE). They are both unconstrained search and optimization algorithms. However, if some constraint handling techniques (CHTs) are incorporated in their frameworks, then they can be used to solve constrained optimization problems (COPs). In an early work, an ensemble of constraint handling techniques (ECHT) is probabilistically hybridized with the basic version of DE. The ECHT consists of four different CHTs: superiority of feasible solutions, self-adaptive penalty, ε-constraint handling technique and stochastic ranking. This paper employs ECHT in the selection schemes, where offspring competes with their parents for survival to the next generation, of JADE and SaDE. As a result, JADE-ECHT and SaDE-ECHT are developed, which are the constrained variants of JADE and SaDE. Both algorithms are tested on 24 COPs and the experimental results are collected and compared according to algorithms' evaluation criteria of CEC'06. Their comparison, in terms of feasibility rate (FR) and success rate (SR), shows that SaDE-ECHT surpasses JADE-ECHT in terms of FR, while JADE-ECHT outperforms SaDE-ECHT in terms of SR.

Keywords: evolutionary algorithms; formal methods in evolutionary algorithms, differential evolution, self-adaptive differential evolutionary algorithms; metaheuristics; mutation strategies; parameters' adaptation; constrained optimization; ensemble of constraint handling techniques; and hybrid algorithms

1. Introduction

Evolutionary algorithms (EAs) are nature inspired population-based stochastic search and optimization methods. EAs work on the principle of natural evolution. In EAs, selected population members based on a fitness/selection scheme, the so called parents, undergo perturbation by applying genetic operators, mutation and crossover, to produce offspring. A selection scheme is then adopted to select the fittest individuals with a certain probability among the parents and offspring for the next generation. Many EAs, such as genetic algorithms (GAs), differential evolution (DE), particle swarm optimization (PSO), firefly algorithm (FA), bee algorithm (BA), ant colony optimization (ACO),

evolution strategy (ES) etc. have been designed using different genetic operators and selection schemes for the unconstrained optimization problems since 1959.

Differential evolution (DE) [1] has proven to be a simple and efficient EA for many optimization problems. A number of variants of DE were developed and are in practice for unconstrained/constrained optimization [2–9]. In DE, a random initial population of size NP is generated in the whole search space to a possible extent and the fittest/best with minimum function value in the initial population is found. It then invokes one of the different mutation strategies such as DE/rand/1, DE/current to best/2, DE/rand/2, DE/current-to-rand/1 to generate a mutant vector. For example in DE/rand/1, the weighted difference scaled by a scaling factor $F \in [0,\ 2]$ between two population vectors is added to a target vector to generate a mutant vector. Afterwards, the parameters of mutant vector and target vector are mixed with a certain crossover probability $C_r \in [0,\ 1]$ to produce the trial vector. Using the one-to-one-spawning selection mechanism, if the objective function value of the trial vector is less than the objective function value of the target vector, in minimization sense, then the trial vector replaces the target vector and becomes the parent for the next generation. The three steps of producing the mutant vector, the trail vector and comparison of the target and trial vectors are repeated until a stopping criterion is met. Also, the fittest/best individual is updated after every generation by comparing the function values of the trial vector, if it is successful in selection process, and fittest/best individual found so far. For more details of DE and different mutation strategies used in it, the readers are referred to [10,11].

The performance of the original DE algorithm is highly dependent on the mutation strategies and its parameters' settings [11–14]. During different evolution stages, different strategies and different parameters' settings with different global and local search capabilities might be preferred. Huang et al. developed a self-adaptive DE variant, SaDE [10]. SaDE automatically adapts the learning strategies and the parameters' settings during evolution. It probabilistically selects one of the four mutation strategies: DE/rand/1, DE/current to best/2, DE/rand/2, DE/current-to-rand/1 for each individual in the current population. J. Zhang and A. C. Sanderson developed another self-adaptive DE version, self-adaptive differential evolution with optional external archive (JADE) [15]. JADE too automates the parameters and employs the mutation strategy DE/current-to-pbest with the optional external archive. The strategy DE/current-to-pbest uses not only the information of the best solution, but also the information on the other good solutions. The external archive keeps record of the inferior solutions, which are then used for diversity among population members and avoiding premature convergence.

For recent advances in DE, the readers are referred to [16,17]. EAs suit a variety of applications in the fields of engineering and science [18–24]. Generally, EAs outperform traditional optimization algorithms for problems which are not continuous, non-differentiable, multi-modal, noisy and not well-defined. However, EAs are unconstrained optimization techniques. They are not capable to directly solve COPs having constraints of any kind (e.g., equality, inequality, linear and non-linear etc.). To overcome this problem, CHTs are used with EAs to handle all types of constraints. The last three decades have witnessed many techniques for handling constraints by EAs [20,25]. Michalewicz and Schoenauer [26] categorized them into five classes: preserving feasibility of solutions, adopting penalty functions, separating feasible solutions from infeasible ones, decoding, and hybridizing different techniques. However, according to no free lunch theorem (NFL) [27], a single CHT can not outperform all other CHTs on each problem. Same is true for different EAs as well. Thus, one has to try and combine different CHTs and EAs to design a suitable algorithm that can solve most of the problems. So keeping in mind the NFL theorem and some other individual problems of COPs, an ensemble of constraint handling techniques (ECHT) is combined with the basic version of DE in [28,29]. ECHT consists of four different CHTs: superiority of feasible solutions, self-adaptive penalty, ε-constraint handling technique and stochastic ranking. SaDE and JADE, being advanced self-adaptive variants, are both unconstrained search and optimization algorithms. Like other EAs, they also need some additional CHTs to solve constrained optimization problems (COPs).

In this work, the ECHT is implemented in the selection scheme, where offspring and parents compete for survival to next generation, of JADE and SaDE. As a result, constrained versions of JADE and SaDE, denoted by JADE-ECHT and SaDE-ECHT, are developed. The performance of JADE-ECHT and SaDE-ECHT is tested and compared based on feasibility rate (FR) and success rate (SR) on 24 COPs according to algorithms' evaluation criteria of CEC'06.

This rest of this paper is ordered as follows. The general COP and ECHT are detailed in Section 2. Section 3 presents the proposed modified algorithms, JADE-ECHT and SaDE-ECHT. Section 4 presents and discusses the experimental results obtained with JADE-ECHT and SaDE-ECHT. Finally, Section 5 describes the concluding remarks of this work.

2. Constrained Optimization Problem and ECHT

This section first describes the constrained optimization problem to be considered in this work. It then illustrates the four CHTs of ECHT.

2.1. Constrained Optimization Problem (COP)

Time, physical, and geometric etc. type constraints exist in most of the real world optimization problems. Such problems can be modelled as a COP. Mathematically, a COP, in case of minimization, can be formulated as follows [30]:

$$\begin{aligned} &\text{Minimize} f(\mathbf{x}),\ \mathbf{x} = [x_1, x_2, \ldots, x_n]^T \\ &\text{subject to} \\ &g_i(\mathbf{x}) \leq 0 \qquad i = 1, \ldots, l, \\ &h_j(\mathbf{x}) = 0 \qquad j = l+1, \ldots, p, \\ &l_i \leq x_i \leq u_i, i = 1, 2, \ldots, n.. \end{aligned} \tag{1}$$

In problem (1), $f(\mathbf{x})$ is called cost function which will be minimized. In case of maximizing the cost function, it needs to be multiplied with a negative sign. The n-dimensional vector $\mathbf{x}$ is called decision variable vector. There are l inequality and $p - l$ equality constraints. An inequality constraint $g_j(\mathbf{x})$ becomes an active constraint, if $g_j(\mathbf{x}^*) = 0$, where $\mathbf{x}^*$ is global optimum solution, whereas equality constraints, $h_j(\mathbf{x}) = 0$, are active by default. Generally, equality constraints are converted into inequality constraints by $|h_j(\mathbf{x})| - \epsilon \leq 0$, where ϵ is an acceptable tolerance for equality constraints. According to CEC'06 [30] evaluation criteria, ϵ is set to 0.0001 (in this work, we will also use the same value for ϵ). l_i and u_i are the lower and upper bounds of component x_i of vector $\mathbf{x}$. They form the whole search space S. The solution $\mathbf{x} \in S$ is referred to be feasible, if it satisfies all the equality and inequality constraints of problem (1); otherwise, it is called infeasible. We denote with F the set of all feasible solutions and normally $F \subset S$. The total constraints' violation for an infeasible solution is defined as [28,29]:

$$v(\mathbf{x}) = \frac{\sum_{i=1}^{p} c_i(g_i'(\mathbf{x}))}{\sum_{i=1}^{p} c_i}, \tag{2}$$

where

$$g_i'(\mathbf{x}) = \begin{cases} max\{g_i(\mathbf{x}), 0\}, & i = 1, \ldots, l \\ max\{|h_j(\mathbf{x})| - \epsilon, 0\}, & j = l+1, \ldots, p. \end{cases}, \tag{3}$$

where $c_i (= 1/g'_{max_i})$ denotes weight parameter, g'_{max_i} denotes the maximum constraint violation of constraint $g_i(\mathbf{x}), i = 1, \ldots, l$ obtained thus far. It maybe noted that c_i changes during the evolution process. This helps in balancing how each constraint contributes in the problem irrespective of their different numerical ranges. The four constraints handling techniques which are used in this work are detailed as follows.

2.2. Superiority of Feasible Solutions (SF)

As the name suggests, in SF feasible solutions have priority over infeasible solutions. SF was first suggested by Deb [31]. In this method, two solutions, a parent $\mathbf{x}^i$ and an offspring $\mathbf{x}^j$ compete. The parent $\mathbf{x}^i$ is considered better than the offspring $\mathbf{x}^j$, if any of the subsequent three settings is met [31]:

- Parent, $\mathbf{x}^i$ is feasible and offspring, $\mathbf{x}^j$ is infeasible.
- Both parent and offspring, $\mathbf{x}^i$ and $\mathbf{x}^j$ are feasible, but parent, $\mathbf{x}^i$ has minimum fitness value than the offspring, $\mathbf{x}^j$.
- Both $\mathbf{x}^i$ and $\mathbf{x}^j$ are infeasible, and overall constraints' violation $v(\mathbf{x}^i)$ of parent, $\mathbf{x}^i$ is less than overall constraints' violation $v(\mathbf{x}^j)$ of offspring, $\mathbf{x}^j$, where $v(\mathbf{x}^i)$ and $v(\mathbf{x}^j)$ are calculated by using Equation (2).

2.3. Self-Adaptive Penalty (SP)

Penalty methods are the most common approaches to handle constraints in the family of CHTs. In these techniques, in order to penalize an infeasible solution, the cost value of each infeasible solution and a penalty term corresponding to its constraints' violation are added, in minimization sense (subtracted in maximization sense). In SP [28], an attempt has been made to facilitate the algorithm to search for feasible solutions, in case there are few feasible solutions, and find the optimum, in case there are enough feasible solutions. For this purpose, two penalties are added to the cost of an infeasible solution. This help in identifying the best infeasible solutions in the existing population. The amount of the added penalties considers the number of feasible solutions that exist in the current population. Thus, if there are few feasible solutions in the combined population of parents and offspring, the amount of penalty to infeasible individuals with higher constraints' violation will be greater. On the contrary, with many feasible solutions, the fittest infeasible solutions in terms of cost are less penalized.

2.4. The ε-Constraint (EC) Handling Technique

The ε-constraint (EC) handling technique [32] adopts the parameter ε to relax the active constraints. The parameter ε is updated until a fixed generation counter is reached. Afterwards, ε becomes 0 to get individuals with no constraints' violation (for detailed formulation of this technique, please see [28,32]).

2.5. Stochastic Ranking (SR)

SR [33] stochastically balances overall constraints' violation and fitness function value. A solution is ranked based on its cost value, if it is feasible or if a randomly generated number is smaller than a probability factor p_f; otherwise, it is ranked on the constraints' violation. The proposed value of $p_f = 0.475$. However, if this constant value is not used, then it decreases linearly from $p_f = 0.475$ to $p_f = 0.025$ from initial generation to the last generation.

In [28], the ECHT is tested with evolutionary programming (EP) and basic DE. In this paper, we hybridize ECHT with the advanced versions of DE, JADE [15] and SaDE [10]

3. JADE-ECHT and SaDE-ECHT

In this section, we first give the algorithmic details of JADE-ECHT, which is then followed by the details of SaDE-ECHT.

3.1. JADE-ECHT

JADE [15] is an updated version of DE. It is also an unconstrained optimization algorithm. So it needs some additional CHTs to solve COPs. In this work, we embed the four above discussed CHTs in the selection scheme of JADE to modify it for solving COPs. The whole procedure of the proposed technique JADE-ECHT, shown in Figure 1, is discussed as follows.

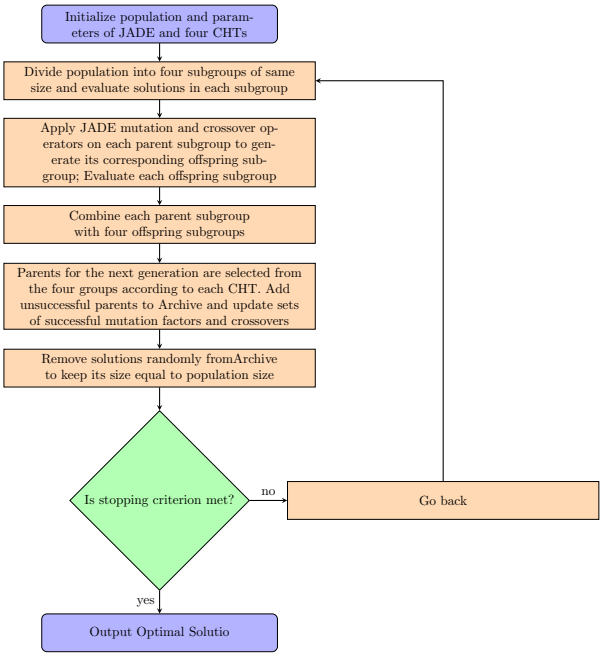


Figure 1. Flowchart of self-adaptive differential evolution with optional external archive (JADE)-ensemble of constraint handling techniques (ECHT).

Step 1: generate initial population P, set the generation number $t = 1$, initial crossover probability $\mu_{CR} = 0.5$, initial mutation factor $\mu_F = 0.5$, the set of archive inferior solutions $A_i = \emptyset$, the sets of successful mutation factors and crossovers, $S_F^i = \emptyset$, $S_{CR}^i = \emptyset$, respectively, where $i = 1, \ldots, 4$.

Step 2: divide population P into four subpopulations, $P_i, i = 1, \ldots, 4$ each of size PS (population size to be tackled by each CHT). Set parameters PAR_i, $i = 1, \ldots, 4$ of PS individuals each with dimension D according to the rules of JADE and corresponding CHT. Also, calculate F_l^i and $CR_l^i, \forall\, l \in \{1, \ldots, PS\}$, where $CR_l^i = randn_l^i\,(\mu_{CR}, 0.1)$, $F_l^i = randn_l^i\,(\mu_{CR}, 0.1)$.

Step 3: compute the cost and the total constraints' violation for every solution in each subpopulation using Equations (1)–(3).

Step 4: each parent subpopulation $(P_i, i = 1, \ldots, 4)$ generates offspring subpopulation $(OFF_i, i = 1, \ldots, 4)$ as a result of applying mutation and crossover operators, respectively as follows [15]:

$$\mathbf{v}_{l,t}^i = \mathbf{x}_{l,t}^i + F_l^i.(\mathbf{x}_{pbest,t}^i - \mathbf{x}_{l,t}^i) + F_l^i.(\mathbf{x}_{r_1^l,t}^i - \widetilde{\mathbf{x}}_{r_2^l,t}^i),$$

where $\mathbf{x}_{pbest,t}$ is one of the $100P\%$ best vectors. and $\widetilde{\mathbf{x}}_{r_2^l,t}^i \neq \mathbf{x}_{r_1^l,t}^i \neq \mathbf{x}_{l,t}^i$ are chosen randomly from the existing population, P and from the union of current population and archived population, $P \cup A$. The archive A retains the parent individuals that are unsuccessful in the selection scheme.

$$u_{l,t}^i = \begin{cases} v_{l,t}^i, & \text{if } (rand_j[0,1] < \mu_{CR} \text{ or } (j = j_{rand}) \\ x_{l,t}^i, & \text{otherwise .} \end{cases} \tag{4}$$

In Equation (4), $v_{l,t}^i$ and $x_{l,t}^i$ are the lth components of the ith mutant and trial vectors in generation t.

Step 5: evaluate the cost and the total constraints' violation for every offspring in each subpopulation using Equations (1)–(3). Every offspring holds the cost and constraints values distinctly.

Step 6: each parent subpopulation is grouped together with its own offspring and the offspring produced by the remaining three subpopulations corresponding to different CHTs. This way four different groups of populations are generated.

Step 7: Parents population P for the next generation is selected from the four groups according to the rule of each CHT. Unsuccessful parents are added to the archive A_i. All successful crossover probabilities from CR^i_{PS} and mutation factors from F^i_{PS} are added to S^i_{CR} and S^i_F.

Step 8: Remove solutions randomly from A_i so that $|A_i| \leq PS$. Update μ_{CR} and μ_F adopting the formulations of [28].

Step 9: If the stopping criteria are not met, go to Step 2; otherwise, stop.

3.2. SaDE-ECHT

SaDE [10] is also an unconstrained optimization algorithm. Like JADE and other EAs, it also needs some additional mechanisms to solve COPs. In this work, the four CHTs of ECHT are used in the selection scheme of SaDE for solving COPs. The whole procedure of proposed SaDE-ECHT, shown in Figure 2, is as follow:

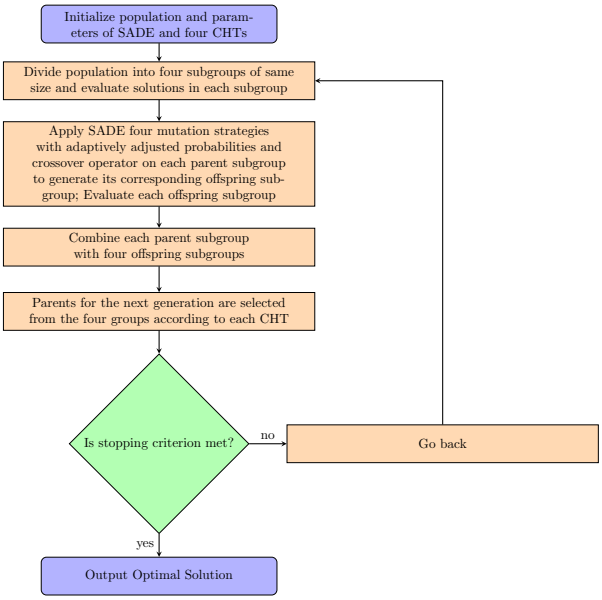


Figure 2. Flowchart of JADE-ECHT.

Step 1: generate initial population P, initiate the generation counter $t = 1$, initial crossover probability $\mu_{CR} = 0.5$, initial mutation factor $\mu_F = 0.5$.

Step 2: divide population P into four subpopulations, $P_i, i = 1, \ldots, 4$ each of size PS. Set parameters PAR_i, where $i = 1, \ldots, 4$ of PS individuals each with dimension D and generate F in [0,2] and CR in (0,1) by using normal distribution according to the rules of SaDE and corresponding CHT.

Step 3: compute the cost and the total constraints' violation for every solution in each subpopulation using Equations (1)–(3).

Step 4: each parent in each subpopulation produces offspring by using one of the four mutation strategies, DE/rand/1, DE/current-to-best/2, DE/rand/2, and DE/current-to-rand/1 (for details of these strategies, please see [10]) and crossover given in Equation (4). For first 20 generations, probabilities are fixed and set to $p_1 = p_2 = p_3 = p_4 = 0.25$. Afterwards, the Roulette Wheel selection is adopted to update the respective probability p_i as follows [10]:

$$p_i = \frac{ns_i}{ns_i + nf_i}, i = 1, 2, 3, 4 \tag{5}$$

Step 5: evaluate the cost and the total constraints' violation for every offspring in each subpopulation using Equations (1)–(3).
Step 6: each parent subpopulation is grouped together with its own offspring and the offspring produced by the remaining three subpopulations corresponding to different CHTs. This way four different groups of populations are generated.
Step 7: parents population P for the next generation are selected from the four groups according to the rule of each CHT.
Step 8: recalculate crossover probability after every five generations according to the mean of recorded CR values.
Step 9: if the stopping criteria are not met, go to Step 2; otherwise, stop.

4. Experimental Results

The performances of JADE-ECHT and SaDE-ECHT were evaluated on the suit of CEC'06, which contains twenty four benchmark functions. The PC configuration and parameters' settings are given in Tables 1 and 2.

Table 1. Configuration of the PC.

System	Windows 8
CPU	3.00 GHz
Ram	2 GB
Language	MATLAB 2012, 8.0.0.783

Table 2. Parameters' settings.

Parameters' Description	Parameters' Settings
Population size for each CHT	$PS = 25$
Whole population size	$NP = 4 * PS = 100$
Maximum number of generations	$t = 2500$
Total number of runs	$runs = 25$
Initial value of mutation factor	$\mu_F = 0.5$
Initial value of crossover probability	$\mu_{CR} = 0.5$
Termination criterion based on maximum function evaluations	$max\_FEs = 500{,}000.$

4.1. Result Achieved

In Tables 3–6, a comparison of both algorithms after 5×10^5 FEs is shown. All the obtained results are gathered according to CEC'06 [30] algorithms' evaluation criteria for problems g01 to g24. The criteria include collecting statistics of the best (minimum), worst (maximum), median, mean and standard deviation of the function error values $f(\mathbf{x}) - f(\mathbf{x}^*)$, where $f(\mathbf{x})$ is the best objective function value obtained by the algorithm after 5×10^5 FEs and $f(\mathbf{x}^*)$ is the know objective function value at the optimal solution. The numbers in parenthesis after the objective function value show the number of violated constraints, whereas c determines the number of violated constraints at the median solution with violation greater than $0.1, 0.001, 0.0001$. $\overline{v}$ shows mean violation at median solution, FR is the feasibility rate which is defined as the number of feasible runs over total runs, and SR is success rate given by the number of successful runs over total runs. A run is called a feasible run, if the algorithm attains in max_FEs at least one feasible solution. Likewise, a run is successful, if the algorithm gets a feasible solution for which the function error value is smaller than 0.0001 in max_FEs.

Table 3 compares the experimental results achieved by JADE-ECHT and SaDE-ECHT for problems g01–g06. This table shows that SaDE-ECHT achieved better statistics in terms of best, median, mean and standard deviation values than JADE-ECHT on problems g01 and g03, whereas JADE-ECHT surpasses SaDE-ECHT on problems g02 and g05 except the best value of g02. It can also be observed from the same table that both algorithms show comparable performance on problems g04 and g06.

The table also shows that both algorithms have achieved 100% FR on all six problems, as can be confirmed from the 0*s* in parenthesis after the objective function values, and columns for c and $\overline{v}$. The SR of SaDE-ECHT on problems g01–g03 is higher than JADE-ECHT. JADE-ECHT's SR is better than SaDE-ECHT on problem g05, while both algorithms obtained the same SR of 100% on problems g04 and g06.

Table 3. Comparison of self-adaptive differential evolution with optional external archive (JADE)-ensemble of constraint handling techniques (ECHT) and self-adaptive differential evolution (SaDE)-ECHT after FES = 500,000 for g01–g06. The bold numbers indicate the better results.

Prob	Algorithm	Best	Median	Worst	c	$\overline{v}$	Mean	Std	FR	SR
g01	JADE-ECHT	0(0)	0(0)	2.0000(0)	0, 0, 0	0	0.0800	0.4000	100%	96%
	SaDE-ECHT	0(0)	0(0)	**0**(**0**)	0, 0, 0	0	**0**	**0**	100%	**100**%
g02	JADE-ECHT	0.0001(0)	**0.0004**(**0**)	**0.0276**(**0**)	0, 0, 0	0	**0.0064**	**0.0091**	100%	16%
	SaDE-ECHT	**0**(**0**)	0.0110(0)	0.1263(0)	0, 0, 0	0	0.0191	0.0254	100%	**24**%
g03	JADE-ECHT	0.0250(0)	0.1015(0)	0.4245(0)	0, 0, 0	0	0.1385	0.1036	100%	0%
	SaDE-ECHT	**0**(**0**)	**0.0122**(**0**)	**0.1524**(**0**)	0, 0, 0	0	**0.0243**	**0.0343**	100%	**12**%
g04	JADE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
	SaDE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
g05	JADE-ECHT	0(0)	**0**(**0**)	**0**(**0**)	0, 0, 0	0	**0**	**0**	100%	**100**%
	SaDE-ECHT	0(0)	91.4773(0)	515.4900(0)	0, 0, 0	0	110.1546	101.2496	100%	4%
g06	JADE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
	SaDE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%

Table 4 presents the experimental statistics achieved by JADE-ECHT and SaDE-ECHT for problems g07–g12. The results of this table show that both algorithms obtained comparable statistics for problems g08, g11 and g12. This table also shows superior performance of SaDE-ECHT in terms median, mean and standard deviation values than JADE-ECHT on the problems g07, g09 and g10 except the best values on problems g07 and g10, where JADE-ECHT got better best values. The table also confirms that both algorithms have achieved 100% FR on all six problems, as can be seen from the 0*s* in parenthesis after the objective function values, and columns for c and $\overline{v}$. The SR of JADE-ECHT on problems g07 and g10 is higher than SaDE-ECHT. SaDE-ECHT's SR is better than JADE-ECHT on problem g09, while both algorithms obtained the same SR of 100% on problems g08, g11 and g12.

Table 5 demonstrates the experimental results achieved by JADE-ECHT and SaDE-ECHT for problems g13–g18. The results of this table show that both algorithms performed similar on problem g16. This table also shows superior performance of SaDE-ECHT in terms best, median, mean and standard deviation values than JADE-ECHT on problems g13 and g18 except the standard deviation of g13, while JADE-ECHT performed better than SaDE-ECHT on problems g14, g15 and g17 except the mean and standard deviation values of problem g14, where SaDE-ECHT got better values for the two quantities. The table also confirms that both algorithms have achieved 100% FR on all six problems, as can be seen from the 0*s* in parenthesis after the objective function values, and columns for c and $\overline{v}$. The SR of JADE-ECHT on problems g14, g15, and g17 is higher than SaDE-ECHT. SaDE-ECHT's SR is better than JADE-ECHT on problems g13 and g18, while both algorithms obtained the same SR of 100% on problem g16.

Table 4. Comparison of JADE-ECHT and SaDE-ECHT after FES = 500,000 for g07–g12. The bold numbers indicate the better results.

Prob	Algorithm	Best	Median	Worst	c	$\overline{v}$	Mean	Std	FR	SR
g07	JADE-ECHT	**0**(**0**)	0.0879(0)	**0.2651**(**0**)	0, 0, 0	0	0.0976	**0.0726**	100%	**4**%
	SaDE-ECHT	0.0001(0)	**0.0114**(**0**)	0.3230(0)	0, 0, 0	0	**0.0518**	0.0850	100%	0%
g08	JADE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
	SaDE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
g09	JADE-ECHT	0(0)	0.0039(0)	0.0714(0)	0, 0, 0	0	0.0132	0.0192	100%	20%
	SaDE-ECHT	0(0)	**0**(**0**)	**0.0006**(**0**)	0, 0, 0	0	**0.0001**	**0.0002**	100%	**76**%
g10	JADE-ECHT	**0**(**0**)	133.9677(0)	343.5425(0)	0, 0, 0	0	143.0809	105.4501	100%	**4**%
	SaDE-ECHT	0.0012(0)	**0.1709**(**0**)	**11.9004**(**0**)	0, 0, 0	0	**1.1748**	**3.0352**	100%	0%
g11	JADE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
	SaDE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
g12	JADE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
	SaDE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%

Table 5. Comparison of JADE-ECHT and SaDE-ECHT after FES = 500,000 for g13–g18. The bold numbers indicate the better results.

Prob	Algorithm	Best	Median	Worst	c	$\overline{v}$	Mean	Std	FR	SR
g13	JADE-ECHT	0.3849(0)	0.9118(0)	0.9459(0)	0, 0, 0	0	0.8275	**0.1750**	100%	0%
	SaDE-ECHT	**0**(**0**)	**0.3870**(**0**)	**0.8491**(**0**)	0, 0, 0	0	**0.3608**	0.2828	100%	**4**%
g14	JADE-ECHT	**0**(**0**)	**0.0174**(**0**)	5.5402(0)	0, 0, 0	0	1.9415	2.2940	100%	**40**%
	SaDE-ECHT	0.4527(0)	1.6397(0)	**3.3912**(**0**)	0, 0, 0	0	**1.7600**	**0.6956**	100%	0%
g15	JADE-ECHT	0(0)	**0**(**0**)	**0**(**0**)	0, 0, 0	0	**0**	**0**	100%	**100**%
	SaDE-ECHT	0(0)	0.0009(0)	2.5449(0)	0, 0, 0	0	0.3333	0.6971	100%	44%
g16	JADE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
	SaDE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
g17	JADE-ECHT	**0**(**0**)	**0**(**0**)	**74.0580**(**0**)	0, 0, 0	0	**8.8870**	**24.5623**	100%	**88**%
	SaDE-ECHT	7.9251(0)	91.2351(0)	297.1687(0)	0, 0, 0	0	92.9967	50.4589	100%	0%
g18	JADE-ECHT	0(0)	0.0001(0)	0.0206(0)	0, 0, 0	0	0.0011	0.0041	100%	52%
	SaDE-ECHT	0(0)	**0**(**0**)	**0**(**0**)	0, 0, 0	0	**0**	**0**	100%	**100**%

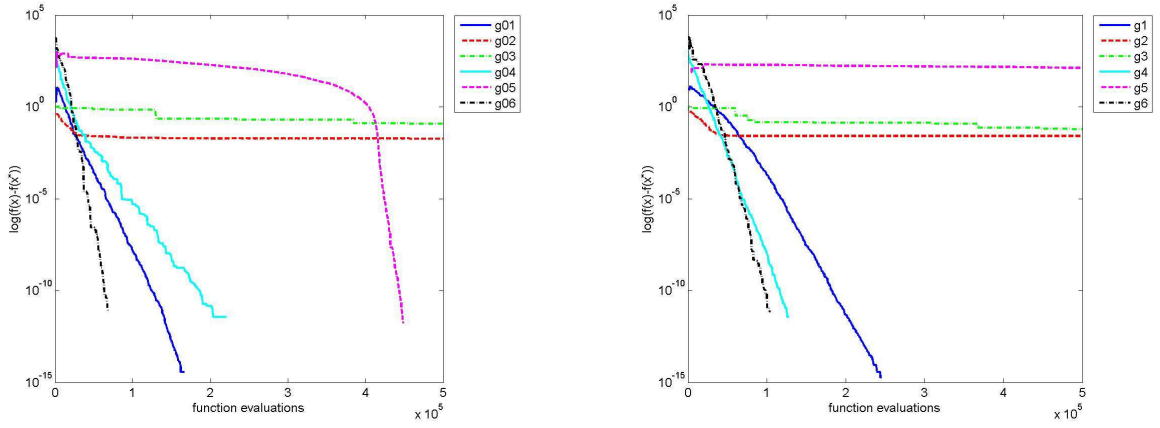
Table 6 presents the experimental results achieved by JADE-ECHT and SaDE-ECHT for problems g19–g24. The results of this table show that both algorithms performed similar on problem g24. This

table also shows superior performance of JADE-ECHT in terms best, median, mean and standard deviation values than SaDE-ECHT on problems g19, g20 g21 and g23, except the best value of problem g20 and standard deviation value of problem g23, while SaDE-ECHT performed better than JADE-ECHT on problem g22. The table also confirms that both algorithms have achieved 100% FR on problems g19 and g24, as can be seen from the 0*s* in parenthesis after the objective function values, and columns for c and $\overline{v}$. Both algorithms are unsuccessful in solving problems g20 and g20. The FR of JADE-ECHT on problem g21 is lower than SaDE-ECHT, while the situation is vice versa in case of SR. The FR and SR of JADE-ECHT on problem g23 is higher than SaDE-ECHT.

Table 6. Comparison of JADE-ECHT and SaDE-ECHT after FES = 500,000 for g19–g24. The bold numbers indicate the better results.

Prob	Algorithm	Best	Median	Worst	c	$\overline{v}$	Mean	Std	FR	SR
g19	JADE-ECHT	**0**(**0**)	**1.4028**(**0**)	**3.6498**(**0**)	0, 0, 0	0	**1.5502**	**1.0136**	100%	**12**%
	SaDE-ECHT	0.3671(0)	1.7022(0)	6.6604(0)	0, 0, 0	0	2.3120	1.9699	100%	0%
g20	JADE-ECHT	3.2029(9)	**6.2057**(**8**)	**15.4062**(**12**)	**1, 1, 2**	**1.1209**	**7.2582**	**3.5087**	0%	0%
	SaDE-ECHT	**2.4461**(**11**)	14.8045(9)	18.3511(11)	2, 4, 4	3.1946	13.1617	4.8304	0%	0%
g21	JADE-ECHT	0(0)	**0.0633**(**0**)	263.7866(1)	0, 0, 0	0	**39.1073**	63.9006	96%	**44**%
	SaDE-ECHT	0(0)	77.3185(0)	**110.2441**(0)	0, 0, 0	0	71.8631	**25.3368**	**100**%	4%
g22	JADE-ECHT	390.4334(4)	10,565.5111(3)	19,715.2233(4)	3, 3, 3	17,5401.6096	10,557.6213	6162.3243	0%	0%
	SaDE-ECHT	**292.6511**(**3**)	**8834.7836**(**3**)	**19,258.8965(3)**	3, 3, 3	**90,196.1317**	**9289.3437**	**4998.2886**	0%	0%
g23	JADE-ECHT	**0**(**0**)	**8.5726**(**0**)	601.1293(0)	0, 0, 0	0	**117.5730**	198.2664	**36**%	**36**%
	SaDE-ECHT	182.7482(0)	357.7081(0)	**518.9083**(**0**)	0, 0, 0	0	344.3397	**87.4764**	0%	0%
g24	JADE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%
	SaDE-ECHT	0(0)	0(0)	0(0)	0, 0, 0	0	0	0	100%	100%

Figure 3 compares the convergence graphs of JADE-ECHT and SaDE-ECHT for problems g01–g06. This figure shows that JADE-ECHT converges faster than SaDE-ECHT on problems g01, g05 and g06, as less number of *FEs* have been used by it. In case of problem g04, the convergence of SaDE-ECHT is speedy than JADE-ECHT, while in case of problems g02, g03 both algorithms converge at the same rate.

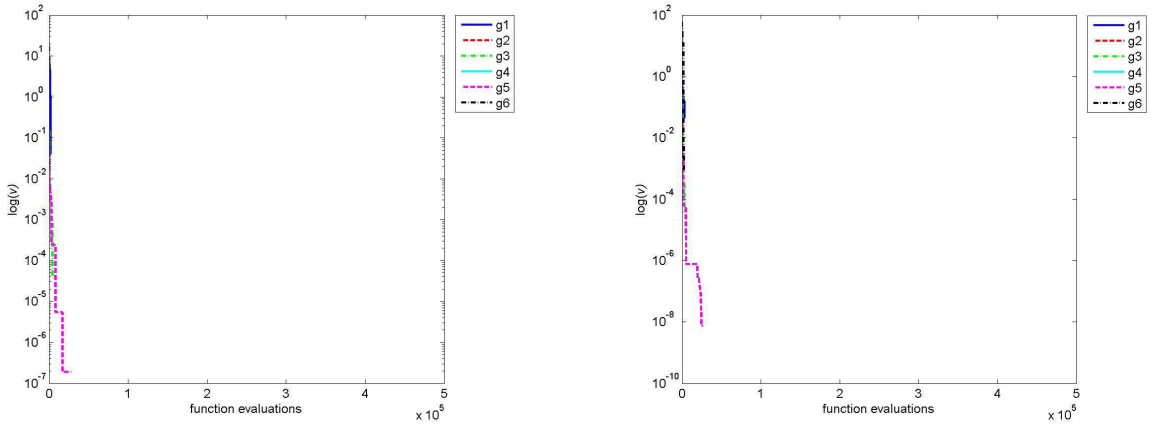


(**a**) JADE-ECHT (**b**) SaDE-ECHT

Figure 3. Convergence comparison of JADE-ECHT and SaDE-ECHT for g01–g06.

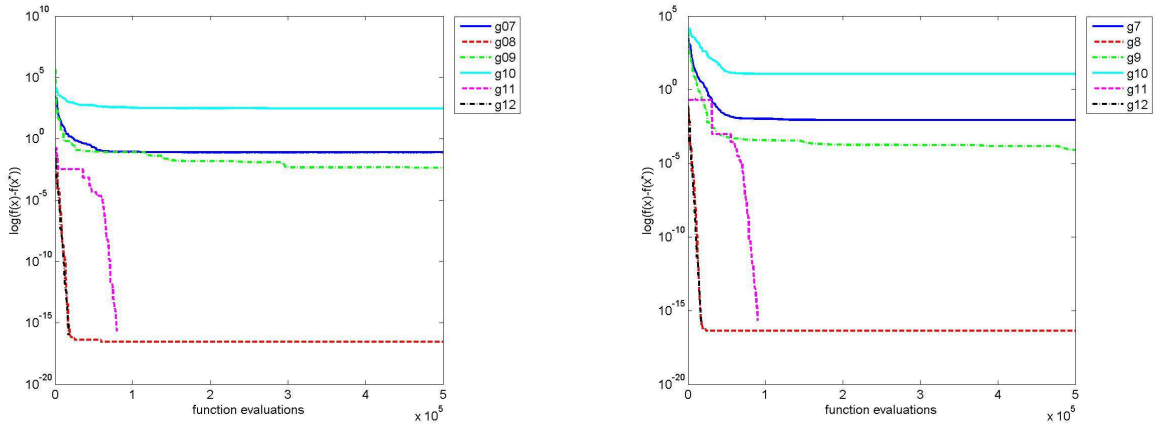
Figure 4 compares the constraints' violations vs *FES* graphs of JADE-ECHT and SaDE-ECHT for problems g01–g06. This figure shows that both algorithms converge quickly to the feasible region and the optimal solution (s) thus has zero constraints' violations.

Figure 5 compares the convergence graphs of JADE-ECHT and SaDE-ECHT for problems g07–g12. This figure shows that both JADE-ECHT and SaDE-ECHT converge at the same rate for all six problems except g11, where JADE-ECHT converges faster than SaDE-ECHT.



(**a**) JADE-ECHT (**b**) SaDE-ECHT

Figure 4. Constraint violation comparison of JADE-ECHT and self-adaptive differential evolution (SaDE)-ECHT for g01–g06.

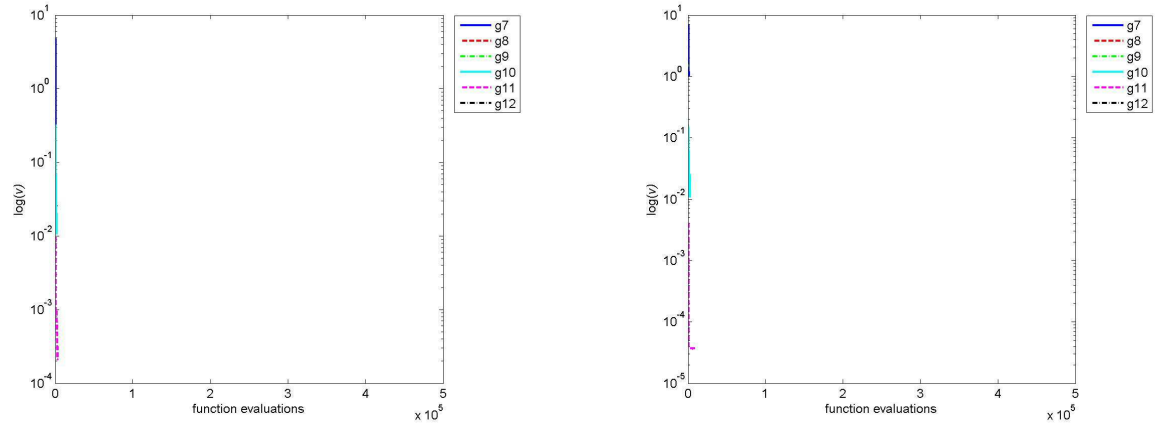


(**a**) JADE-ECHT (**b**) SaDE-ECHT

Figure 5. Convergence comparison of JADE-ECHT and SaDE-ECHT for g07–g12.

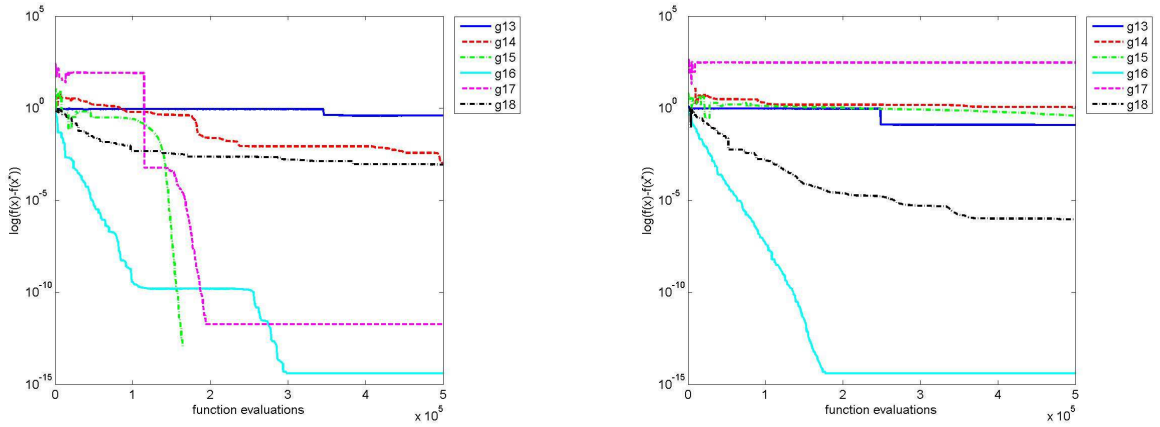
Figure 6 compares the constraints' violations vs *FES* graphs of JADE-ECHT and SaDE-ECHT for problems g07–g12. This figure too shows that both algorithms converge quickly to the feasible region and optimal solution(s) thus has zero constraints' violations.

Figure 7 compares the convergence graphs of JADE-ECHT and SaDE-ECHT for problems g13–g18. This figure shows that both JADE-ECHT and SaDE-ECHT converge at the same rate for all six problems except g15, where JADE-ECHT converges faster than SaDE-ECHT.



(**a**) JADE-ECHT (**b**) SaDE-ECHT

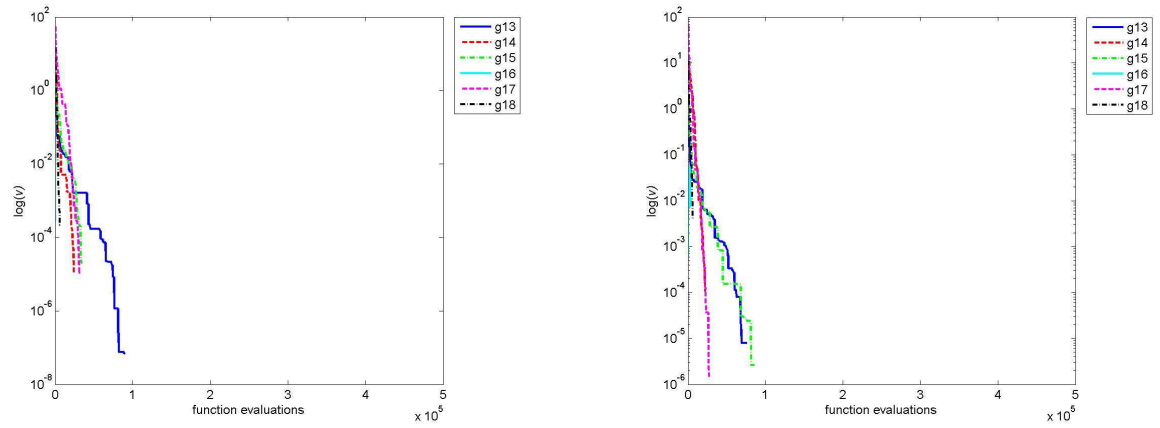
Figure 6. Constraint violation comparison of JADE-ECHT and SaDE-ECHT for g07–g12.



(**a**) JADE-ECHT (**b**) SaDE-ECHT

Figure 7. Convergence comparison of JADE-ECHT and SaDE-ECHT for g13–g18.

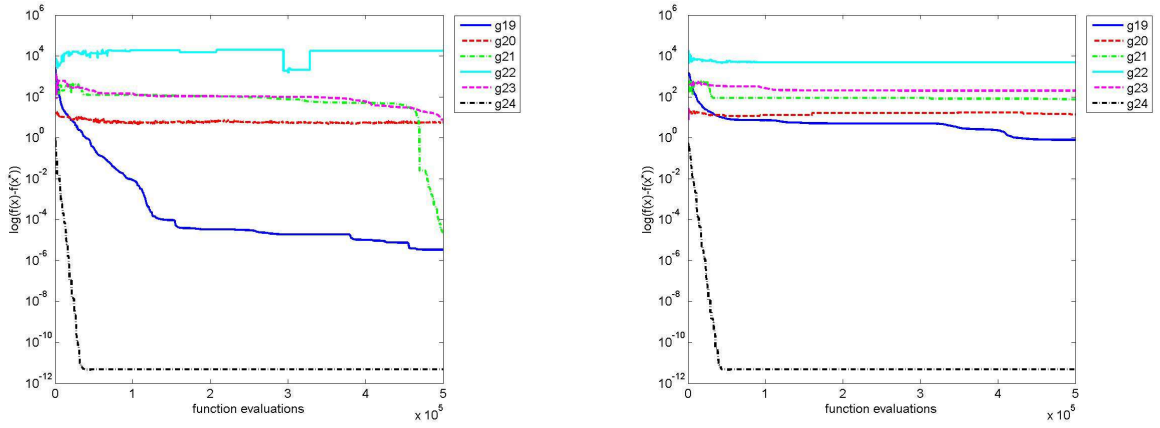
Figure 8 compares the constraints' violations vs *FES* graphs of JADE-ECHT and SaDE-ECHT for problems g13–g18. This figure shows that both algorithms explore the infeasible region for about 1000 iterations and then converge to the feasible region. As a result, optimal solution(s) thus obtained has zero constraints' violations.



(**a**) JADE-ECHT (**b**) SaDE-ECHT

Figure 8. Constraint violation comparison of JADE-ECHT and SaDE-ECHT for g13–g18.

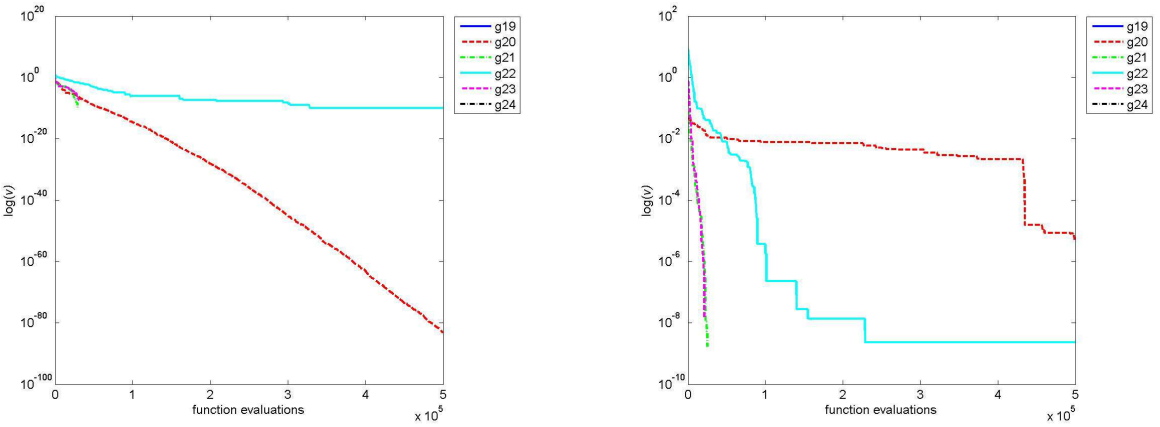
Figure 9 compares the convergence graphs of JADE-ECHT and SaDE-ECHT for problems g19–g24. This figure shows that both JADE-ECHT and SaDE-ECHT converge almost at the same rate for all six problems and utilize the maximum function evaluations.



(**a**) JADE-ECHT (**b**) SaDE-ECHT

Figure 9. Convergence comparison of JADE-ECHT and SaDE-ECHT for g19–g24.

Figure 10 compares the constraints' violations vs *FES* graphs of JADE-ECHT and SaDE-ECHT for problems g19–g24. This figure clearly shows that both algorithms failed to obtain any feasible solution in case of problems g20 and g22, although maximum function evaluations have been used.



(**a**) JADE-ECHT (**b**) SaDE-ECHT

Figure 10. Constraint violation comparison of JADE-ECHT and SaDE-ECHT for g19–g24.

Figures 3,5,7 and 9 show the comparison of the convergence graphs vs *FES* of both algorithms for all problems g01-g24, whereas Figures 4,6,8 and 10 demonstrate their comparison graphs of the constraints' violations vs *FEs*.

Overall, it can be concluded from the tabulated results and figures that both algorithms have achieved feasible solution (s) and near optimal solution (s) on 22 problems out of 24 except problems g20 and g22. The tables show that the FR of JADE-ECHT on 20 problems out of 24 is 100% and that of SaDE-ECHT on 22 problems out of 24 is 100%. The SR of JADE-ECHT on most of the problems is better than SaDE-ECHT. On two problems g20 and g22, the FR and SR of both algorithms are 0%. The dimension of these two problems is higher than other 22 problems. Also, these two problems had a large number of equality constraints. It can be noted from our experiments and some other literature review that equality constraints were hard to handle.

Table 7 compares the FR and SR of JADE-ECHT and SaDE-ECHT with other competing algorithms of CEC'2006. It can be seen from the said table that both JADE-ECHT and SaDE-ECHT achieved better FR, and can be placed at positions second and fourth, respectively. However, they failed to achieve better SR than the competing algorithms. A reason of failure could be the use of four different CHTs, where the resources (FEs) are distributed based on the success of each individual CHT, while the competing algorithms used just one CHT. The same can also be observed from Tables 8 and 9, where the median and standard deviation values obtained after 5×10^5 FEs of JADE-ECHT and SaDE-ECHT are compared with other competing algorithms (the values of the two quantities for the competing algorithms are taken from each source paper). Another reason of low SR could be observed from the figures showing constraints' violations vs FES graphs. It can be noticed from these graphs that both algorithms converge quickly to the feasible region. As a result, they less explore the infeasible region and consequently suffer from stagnation and premature convergence.

Table 7. Comparison of JADE-ECHT and SaDE-ECHT in terms of feasibility rate (FR) and success rate (SR) with algorithms of CEC 2006.

Algorithms	FR	SR
DE	95.65%	78.09%
DMS-PSO	100%	90.61%
ϵ DE	100%	95.65%
GDE	92.00%	77.39%
jDE-2	95.65%	80.00%
MDE	95.65%	87.65%
MPDE	94.96%	87.65%
PCX	95.65%	94.09%
PESO+	95.48%	67.83%
SaDE	100%	87.13%
JADE-ECHT	95.30%	57.04%
SaDE-ECHT	95.65%	46.43%

Table 8. Comparison of median values of JADE-ECHT, SaDE-ECHT and CEC'2006 algorithms achieved after 500,000 FEs. The bold numbers indicate the better results.

Prob	DE	DMS-PSO	ϵ DE	GDE	jDE-2	MDE	MPDE	PCX	PESO+	SaDE	JADE-ECHT	SaDE-ECHT
g01	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
g02	5.1700×10^{-8} (0)	**0(0)**	3.0933×10^{-8}(0)	2.3251×10^{-7}(0)	3.3051×10^{-9}(0)	0.017460(0)	3.5608×10^{-6}	**0(0)**	1.4314×10^{-6}(0)	3.0800×10^{-9}(0)	0.0004(0)	0.0110(0)
g03	6.7110×10^{-1}(0)	**0(0)**	-4.4409×10^{-16}(0)	9.3634×10^{-1}(0)	0.3481(0)	**0(0)**	-2.8866×10^{-15}	**0(0)**	1.5890×10^{-7} (0)	1.7770×10^{-8}(0)	0.1015(0)	0.0122(0)
g04	7.6398×10^{-11}(0)	**0(0)**	**0(0)**	8.0036×10^{-11}(0)	**0(0)**	**0(0)**	3.6380×10^{-12}	**0(0)**	1.0000×10^{-10}(0)	2.1667×10^{-7}(0)	**0(0)**	**0(0)**
g05	-9.0949×10^{-13}(0)	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	91.4773(0)
g06	4.5475×10^{-11}	**0(0)**	1.1823×10^{-11}(0)	6.1846×10^{-11}(0)	1.1823×10^{-11}(0)	**0(0)**	1.0914×10^{-11}	**0(0)**	1.0000×10^{-10}(0)	4.5475×10^{-11}(0)	**0(0)**	**0(0)**
g07	7.9783×10^{-11}(0)	**0(0)**	-1.8474×10^{-13}(0)	3.6402×10^{-10}(0)	-1.8829×10^{-13}(0)	**0(0)**	-1.8474×10^{-13}	**0(0)**	9.4367×10^{-6}(0)	1.4608×10^{-7}(0)	0.0879(0)	0.0114(0)
g08	8.1964×10^{-11}(0)	**0(0)**	4.1633×10^{-17}(0)	8.1964×10^{-11}(0)	4.1633×10^{-17}(0)	**0(0)**	4.1633×10^{-17}	**0(0)**	1.0000×10^{-10}(0)	8.1964×10^{-11}(0)	**0(0)**	**0(0)**
g09	-9.8112×10^{-11}(0)	**0(0)**	**0(0)**	-9.7884×10^{-11}(0)	2.2737×10^{-13}(0)	**0(0)**	1.1369×10^{-13}	**0(0)**	1.0000×10^{-10}(0)	3.7440×10^{-7}(0)	0.0039(0)	**0(0)**
g10	6.2755×10^{-11}(0)	1.0124×10^{-8}(0)	-9.0949×10^{-13}(0)	6.9122×10^{-11}(0)	-9.0949×10^{-13}(0)	**0(0)**	-9.0949×10^{-13}	**0(0)**	1.3432×10^{-3}(0)	1.8120×10^{-6}(0)	133.9677(0)	0.1709(0)
g11	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
g12	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
g13	3.8486×10^{-1}(0)	**0(0)**	9.7145×10^{-17}(0)	3.8486×10^{-1}(0)	0.6800(0)	**0(0)**	3.8486×10^{-1}	**0(0)**	1.1200×10^{-8}(0)	4.1898×10^{-11}(0)	0.9118(0)	0.3870(0)
g14	8.5123×10^{-12}(0)	**0(0)**	2.1316×10^{-14}(0)	6.3148×10^{-8}(0)	2.1316×10^{-14}(0)	**0(0)**	-3.9961×10^{-4}	**0(0)**	3.2912×10^{-3}(0)	1.4793×10^{-5}(0)	0.0174(0)	1.6397(0)
g15	6.0822×10^{-11}(0)	**0(0)**	**0(0)**	6.0936×10^{-11}(0)	**0(0)**	**0(0)**	**0(0)**	**0(0)**	1.0000×10^{-10}(0)	6.0822×10^{-11}(0)	**0(0)**	0.0009(0)
g16	6.5214×10^{-11}(0)	**0(0)**	4.4409×10^{-15}(0)	6.5216×10^{-11}(0)	5.1070×10^{-15}(0)	**0(0)**	5.3291×10^{-15}	**0(0)**	1.0000×10^{-10}(0)	6.5214×10^{-11}(0)	**0(0)**	**0(0)**
g17	7.4058×10^{1}(0)	7.4058×10^{1}(0)	1.8190×10^{-12}(0)	7.4052×10^{1}(0)	10.4896(0)	**0(0)**	7.4058×10^{1}	**0(0)**	13.9638(0)	7.4058×10^{1}(0)	**0(0)**	91.2351(0)
g18	1.5561×10^{-11}(0)	0(0)	3.3307×10^{-16}(0)	4.6362×10^{-11}(0)	4.4408×10^{-16}(0)	**0(0)**	4.4409×10^{-16}	**0(0)**	4.0000×10^{-10}(0)	1.5561×10^{-11}(0)	0.0001(0)	**0(0)**
g19	4.6370×10^{-11}(0)	**0(0)**	5.2162×10^{-8}(0)	5.2669×10^{-9}(0)	4.2632×10^{-14}(0)	0.387033(0)	3.5527×10^{-14}	**0(0)**	2.6302×10^{-2}(0)	1.3868×10^{-10}(0)	1.4028(0)	1.7022(0)
g20	-2.4674×10^{-2}(8)	-7.3330×10^{-2}(17)	-2.4674×10^{-2}(8)	1.3503×10^{1}(20)	0.1082(2)	0.103314(20)	1.0151×10^{1}	0.0675(12)	3.2600×10^{-2}(8)	2.3757×10^{-1}(20)	6.2057(8)	14.8045(9)
g21	-2.8371×10^{-10}(0)	3.5911×10^{-6}(0)	-2.8422×10^{-14}(0)	6.5523×10^{-8}(0)	-2.8421×10^{-14}(0)	**0(0)**	1.4211×10^{-13}	**0(0)**	81.3460(0)	2.5785×10^{-8}(0)	0.0633(0)	77.3185(0)
g22	1.0336×10^{4}(15)	1.2200×10^{2} (0)	1.2332×10^{1}(0)	9.7885×10^{3}(19)	8033.6537(8)	9210.082460(18)	8.7919×10^{3}	9888.6409(15)	14,198.8059(19)	4.6907×10^{1}(0)	10,565.5111(3)	8834.7836(3)
g23	3.0005×10^{2}(0)	1.0267×10^{-8} (0)	**0(0)**	1.0569×10^{1}(0)	2.2737×10^{-13}(0)	**0(0)**	**0(0)**	**0(0)**	130.5043(0)	3.9790×10^{-13}(0)	8.5726(0)	357.7081(0)
g24	4.6736×10^{-12}(0)	**0(0)**	5.7732×10^{-14}(0)	4.7269×10^{-12}(0)	5.5067×10^{-14}(0)	**0(0)**	7.1054×10^{-14}	**0(0)**	**0(0)**	4.6372×10^{-12}(0)	**0(0)**	**0(0)**

Table 9. Comparison of standard deviation values of JADE-ECHT, SaDE-ECHT and CEC'2006 algorithms achieved after 500,000 FEs. The bold numbers indicate the better results.

Prob	DE	DMS-PSO	ϵ DE	GDE	jDE-2	MDE	MPDE	PCX	PESO+	SaDE	JADE-ECHT	SaDE-ECHT
g01	6.4146×10^{-15}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0.4000	0
g02	1.0015×10^{-18}	4.6953×10^{-3}	1.7523×10^{-8}	7.5112×10^{-3}	3.0488×10^{-3}	**0**	2.7802×10^{-3}	**0**	7.1385×10^{-3}	4.9786×10^{-3}	0.0091	0.0254
g03	5.2098×10^{-14}	**0**	2.9582×10^{-31}	1.9833×10^{-1}	1.0140×10^{-1}	**0**	8.3681×10^{-2}	**0**	1.5396×10^{-6}	3.4743×10^{-5}	0.1036	0.0343
g04	3.9644×10^{-13}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1.8550×10^{-12}	**0**	**0**
g05	**0**	**0**	**0**	1.6854×10^{2}	2.4193	**0**	3.6380×10^{-13}	**0**	**0**	1.8190×10^{-13}	0	101.2496
g06	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
g07	6.4146×10^{-15}	**0**	2.1831×10^{-15}	3.8948×10^{-7}	1.7405×10^{-15}	**0**	3.0215×10^{-15}	**0**	2.2678×10^{-5}	1.4993×10^{-5}	0.0726	0.0850
g08	1.0015×10^{-18}	**0**	1.2326×10^{-32}	**0**	1.2580×10^{-32}	**0**	**0**	**0**	**0**	3.8426×10^{-18}	**0**	**0**
g09	5.2098×10^{-14}	**0**	**0**	4.9555×10^{-14}	4.2538×10^{-14}	**0**	**0**	**0**	**0**	7.9851×10^{-14}	0.0192	0.0002
g10	3.9644×10^{-13}	8.8141×10^{-9}	4.2426×10^{-13}	**0**	9.1279×10^{-8}	**0**	3.0165×10^{-13}	**0**	5.8279×10^{-2}	1.5244×10^{-6}	105.4501	3.0352
g11	**0**	**0**	**0**	**0**	4.5540×10^{-4}	**0**	7.0638×10^{-5}	**0**	**0**	**0**	**0**	**0**
g12	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
g13	**0**	1.8204×10^{-6}	0	3.8650×10^{-1}	2.2376×10^{-1}	**0**	2.8719×10^{-1}	**0**	6.3801×10^{-6}	2.7832×10^{-7}	0.1750	0.2828
g14	3.4809×10^{-15}	**0**	1.3924×10^{-15}	3.8556×10^{-3}	3.4809×10^{-15}	**0**	7.9441×10^{-15}	**0**	2.8553×10^{-2}	6.4986×10^{-5}	2.2940	0.6956
g15	**0**	**0**	**0**	9.6094×10^{-1}	2.2020×10^{-2}	**0**	4.3027×10^{-6}	**0**	**0**	**0**	**0**	0.6971
g16	2.2092×10^{-16}	**0**	1.5777×10^{-30}	1.7764×10^{-16}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
g17	3.0234×10^{1}	**0**	1.2117×10^{-27}	8.2114×10^{1}	3.8319×10^{1}	**0**	3.4111×10^{1}	**0**	42.286	1.6168×10^{1}	24.5623	50.4589
g18	4.8817×10^{-17}	**0**	2.1756×10^{-17}	6.4619×10^{-1}	3.6822×10^{-17}	**0**	4.1541×10^{-17}	**0**	3.8184×10^{-2}	5.2898×10^{-2}	0.0041	**0**
g19	1.2568×10^{-5}	**0**	1.2568×10^{-5}	4.5735×10^{-5}	1.0531×10^{-13}	0.847517	3.5527×10^{-14}	**0**	1.6158×10^{-1}	7.2976×10^{-10}	1.0136	1.9699
g20	4.2362×10^{-2}	6.9516×10^{-3}	4.2362×10^{-2}	1.9580×10^{0}	1.1510×10^{-2}	0.021688	2.8984×10^{0}	0.0219	**4.9951** $\times 10^{-3}$	1.0638×10^{-1}	3.5087	4.8304
g21	6.5489×10^{1}	2.0784×10^{-6}	3.3417×10^{-14}	8.6788×10^{1}	3.6266×10^{1}	0	6.2358×10^{1}	**0**	67.019	1.5058×10^{-3}	63.9006	25.3368
g22	5.7875×10^{3}	2.7703×10^{1}	**1.5690** $\times 10^{1}$	5.9865×10^{3}	5.1748×10^{3}	4808.800969	4.8022×10^{3}	4421.5326	4963.7	3.0415×10^{1}	6162.3243	4998.2886
g23	8.8097×10^{-6}	4.5997×10^{-4}	1.1139×10^{-14}	1.9119×10^{2}	5.9983×10^{1}	**0**	5.1159×10^{-14}	**0**	87.634	3.4116×10^{-4}	198.2664	87.4764
g24	8.8525×10^{-20}	**0**	2.5244×10^{-29}	**0**	1.9323×10^{-29}	**0**	**0**	**0**	**0**	**0**	**0**	**0**

5. Conclusions and Future Work

This paper employed ECHT in the frameworks of two self-adaptive variants of DE, JADE and SaDE. Thus, constrained versions of the two algorithms, denoted by JADE-ECHT and SaDE-ECHT were developed. The proposed algorithms JADE-ECHT and SaDE-ECHT were tested and compared on CEC'06 benchmark test suit. The experimental results show that the SR of JADE-ECHT on most of the tested problems is better than SaDE-ECHT, while SaDE-ECHT surpasses JADE-ECHT in terms of FR. Both algorithms, like other algorithms in the literature, failed to solve problems g20 and g22 due to the hard nature of these problems. In the future, we intend to design ECHT of some other CHTs, embed it then in DE and swarm based algorithms to develop constrained evolutionary algorithms and finally test these newly developed algorithms on some real-world and engineering optimization problems. In addition to that we are going to use [34] for multipath routing protocols and for video streaming systems [35] in order to get the advantages of these plus the benefits of the proposed work would be very beneficent and demanded.

Author Contributions: Conceptualization, H.J. and M.A.J.; methodology, H.K., M.A.J. and W. K.M.; software, R.A.K., N.T. and H.S.; validation, H.U.K. and M.S.; formal analysis, H.J., M.A.J., R.A.K. and H.S.; investigation, H.J., M.A.J. and M.S.; resources, N.T. and H.S.; writing—original draft preparation, H.J. and M.A.J.; writing—review and editing, H.U.K. and M.S.; supervision, M.A.J. and W.K.M.; project administration, N.T. and H.S.; funding acquisition, N.T. and H.S.

Funding: The authors would like to thank King Khalid University of Saudi Arabia for supporting this research under the grant number R.G.P.2/7/38.

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

References

1. 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]
2. Li, Z.; Shang, Z.; Liang, J.J.; Niu, B. An improved differential evolution for constrained optimization with dynamic constraint-handling mechanism. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation (CEC), Brisbane, Australia, 10–15 June 2012; pp. 1–6.
3. Elsayed, S.M.; Sarker, R.A.; Essam, D.L. An improved self-adaptive differential evolution algorithm for optimization problems. *IEEE Trans. Ind. Inform.* **2013**, *9*, 89–99. [CrossRef]
4. Li, G.; Lin, Q.; Cui, L.; Du, Z.; Liang, Z.; Chen, J.; Lu, N.; Ming, Z. A novel hybrid differential evolution algorithm with modified CoDE and JADE. *Appl. Soft Comput.* **2016**, *47*, 577–599. [CrossRef]
5. Ali, M.; Kajee-Bagdadi, Z. A local exploration-based differential evolution algorithm for constrained global optimization. *Appl. Math. Comput.* **2009**, *208*, 31–48. [CrossRef]
6. Ameca-Alducin, M.Y.; Mezura-Montes, E.; Cruz-Ramírez, N. Dynamic differential evolution with combined variants and a repair method to solve dynamic constrained optimization problems: an empirical study. *Soft Comput.* **2018**, *22*, 541–570. [CrossRef]
7. Shah, T.; JAN, M.; Mashwani, W.K.; Wazir, H. Adaptive Differential Evolution for Constrained Optimization Problems. *Sci. Int.* **2016**, *28*, 2313–2320.
8. Wazir, H.; Jan, M.; Mashwani, W.; Shah, T. A penalty function based differential evolution algorithm for constrained optimization. *Nucleus* **2016**, *53*, 155–166.
9. Jan, M.A.; Khanum, R.A.; Tairan, N.M.; Mashwani, W.K. Performance of a Constrained Version of MOEA/D on CTP-series Test Instances. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 496–505.
10. Brest, J.; Zumer, V.; Maucec, M.S. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 215–222.
11. Caraffini, F.; Kononova, A.V.; Corne, D. Infeasibility and structural bias in Differential Evolution. *arXiv* **2019**, arXiv:1901.06153.
12. Caraffini, F.; Kononova, A.V. Structural Bias in Differential Evolution: a preliminary study. *AIP Conf. Proc.* **2019**, *2070*, 020005.

13. Yaman, A.; Iacca, G.; Caraffini, F. A comparison of three differential evolution strategies in terms of early convergence with different population sizes. *AIP Conf. Proc.* **2019**, *2070*, 020002.
14. Iacca, G.; Neri, F.; Caraffini, F.; Suganthan, P.N. A differential evolution framework with ensemble of parameters and strategies and pool of local search algorithms. In Proceedings of the European Conference on the Applications of Evolutionary Computation, Granada, Spain, 23–25 April 2014; pp. 615–626.
15. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
16. Caraffini, F.; Neri, F. A study on rotation invariance in differential evolution. *Swarm Evol. Comput.* **2018**. [CrossRef]
17. Caraffini, F.; Neri, F. Rotation invariance and rotated problems: An experimental study on differential evolution. In Proceedings of the International Conference on the Applications of Evolutionary Computation, Parma, Italy, 4–6 April 2018; pp. 597–614.
18. Liu, C.; Wang, G.; Xie, Q.; Zhang, Y. Vibration sensor-based bearing fault diagnosis using ellipsoid-ARTMAP and differential evolution algorithms. *Sensors* **2014**, *14*, 10598–10618. [CrossRef] [PubMed]
19. Datta, R.; Deb, K.; Kim, J.H. CHIP: Constraint Handling with Individual Penalty approach using a hybrid evolutionary algorithm. *Neural Comput. Appl.* **2018**, 1–17. [CrossRef]
20. Shakibayifar, M.; Hassannayebi, E.; Mirzahossein, H.; Taghikhah, F.; Jafarpur, A. An intelligent simulation platform for train traffic control under disturbance. *Int. J. Model. Simul.* **2019**, *39*, 135–156. [CrossRef]
21. Cheraitia, M.; Haddadi, S.; Salhi, A. Hybridizing plant propagation and local search for uncapacitated exam scheduling problems. *Int. J. Serv. Oper. Manag.* **2017**. [CrossRef]
22. Wang, G.G.; Tan, Y. Improving metaheuristic algorithms with information feedback models. *IEEE Trans. Cybern.* **2017**, *49*, 542–555. [CrossRef]
23. Fister, I.; Fister, I., Jr. *Adaptation and Hybridization in Computational Intelligence*; Springer: Berlin, Germany, 2015; Volume 18.
24. Wang, H.; Yi, J.H. An improved optimization method based on krill herd and artificial bee colony with information exchange. *Memet. Comput.* **2018**, *10*, 177–198. [CrossRef]
25. Coit, D.W.; Smith, A.E.; Tate, D.M. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS J. Comput.* **1996**, *8*, 173–182. [CrossRef]
26. Michalewicz, Z.; Schoenauer, M. Evolutionary algorithms for constrained parameter optimization problems. *Evol. Comput.* **1996**, *4*, 1–32. [CrossRef]
27. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
28. Mallipeddi, R.; Suganthan, P.N. Ensemble of constraint handling techniques. *IEEE Trans. Evol. Comput.* **2010**, *14*, 561–579. [CrossRef]
29. Mallipeddi, R.; Suganthan, P.N. Differential evolution with ensemble of constraint handling techniques for solving CEC 2010 benchmark problems. In Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain, 18–23 July 2010; pp. 1–8.
30. Liang, J.; Runarsson, T.P.; Mezura-Montes, E.; Clerc, M.; Suganthan, P.N.; Coello, C.C.; Deb, K. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *J. Appl. Mech.* **2006**, *41*, 8–31.
31. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [CrossRef]
32. Takahama, T.; Sakai, S. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain, 18–23 July 2010; pp. 1–9.
33. Runarsson, T.P.; Yao, X. Stochastic ranking for constrained evolutionary optimization. *IEEE Trans. Evol. Comput.* **2000**, *4*, 284–294. [CrossRef]

34. Iqbal, Z.; Khan, S.; Mehmood, A.; Lloret, J.; Alrajeh, N.A. Adaptive cross-layer multipath routing protocol for mobile ad hoc networks. *J. Sens.* **2016**, *2016*, 5486437. [CrossRef]
35. Taha, M.; Garcia, L.; Jimenez, J.M.; Lloret, J. SDN-based throughput allocation in wireless networks for heterogeneous adaptive video streaming applications. In Proceedings of the 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 26–30 June 2017; pp. 963–968.