mathematics

MDPI

Article

Settings-Free Hybrid Metaheuristic General Optimization Methods

Héctor Migallón [1,*,†], Akram Belazi [2,†], José-Luis Sánchez-Romero [3,†], Héctor Rico [3,†] and Antonio Jimeno-Morenilla [3,†]

[1] Department of Computer Engineering, Miguel Hernández University, E-03202 Elche, Alicante, Spain
[2] Laboratory RISC-ENIT (LR-16-ES07), Tunis El Manar University, Tunis 1002, Tunisia; akram.belazi@enit.utm.tn
[3] Department of Computer Technology, University of Alicante, E-03071 Alicante, Spain; sanchez@dtic.ua.es (J.-L.S.-R.); hector.rico@gmail.com (H.R.); jimeno@dtic.ua.es (A.J.-M.)
* Correspondence: hmigallon@umh.es; Tel.: +34-966658390
† These authors contributed equally to this work.

Received: 2 June 2020; Accepted: 1 July 2020; Published: 3 July 2020

Abstract: Several population-based metaheuristic optimization algorithms have been proposed in the last decades, none of which are able either to outperform all existing algorithms or to solve all optimization problems according to the No Free Lunch (NFL) theorem. Many of these algorithms behave effectively, under a correct setting of the control parameter(s), when solving different engineering problems. The optimization behavior of these algorithms is boosted by applying various strategies, which include the hybridization technique and the use of chaotic maps instead of the pseudo-random number generators (PRNGs). The hybrid algorithms are suitable for a large number of engineering applications in which they behave more effectively than the thoroughbred optimization algorithms. However, they increase the difficulty of correctly setting control parameters, and sometimes they are designed to solve particular problems. This paper presents three hybridizations dubbed HYBPOP, HYBSUBPOP, and HYBIND of up to seven algorithms free of control parameters. Each hybrid proposal uses a different strategy to switch the algorithm charged with generating each new individual. These algorithms are Jaya, sine cosine algorithm (SCA), Rao's algorithms, teaching-learning-based optimization (TLBO), and chaotic Jaya. The experimental results show that the proposed algorithms perform better than the original algorithms, which implies the optimal use of these algorithms according to the problem to be solved. One more advantage of the hybrid algorithms is that no prior process of control parameter tuning is needed.

Keywords: hybrid optimization algorithms; SCA algorithm; jaya; 2D chaotic map; TLBO; Rao's algorithms

1. Introduction

It is well known that metaheuristic optimization methods are widely used to solve problems in several fields of science and engineering. Population-based metaheuristic methods iteratively generate new populations to increase diversity in the current generation. This increases the probability of reaching the optimum for the considered problem. These algorithms are proposed to replace exact optimization algorithms when they are not able to reach an acceptable solution. The inability to provide an adequate solution may be due to either the characteristics of the objective function or the wide search space, which renders a comprehensive search useless. In addition, classical optimization methods, such as greedy-based algorithms, need to consider several assumptions that make it hard to resolve the considered problem.

When metaheuristic methods are operated, on the one hand, the objective function has no restrictions. On the other hand, each optimization method proposes its own rules for the evolution of the population towards the optimum. These algorithms are suitable for general problems, but each one has different skills in global exploration and local exploitation.

Some of the proposed algorithms that have proven to be effective in several areas of science and engineering are studied: mine blast algorithm (MBA) [1] based on the mine bomb explosion concept; the manta ray foraging optimization method (MRFO) [2] based on intelligent behaviors of manta ray; the crow search algorithm (CSA) [3] based on the behavior of crows; the ant colony optimization (ACO) algorithm [4] which imitates the foraging behavior of ant colonies; the biogeography-based optimization (BBO) algorithm [5] which improves solutions stochastically and iteratively; the grenade explosion method (GEM) algorithm [6] based on the characteristics of the explosion of a grenade; the particle swarm optimization (PSO) algorithm [7] based on the social behavior of fish schooling or bird flocking; the firefly (FF) algorithm [8] inspired by the flashing behavior of fireflies; the artificial bee colony (ABC) algorithm [9] inspired by the foraging behavior of honey bees; the gravitational search algorithm (GSA) [10] based on Newton's law of gravity; and the shuffled frog leaping (SFL) algorithm [11] which imitates the collaborative behavior of frogs; among others. Many of them require configuration parameters that must be correctly tuned according to the problem to be solved, see for example [12]. Otherwise, exploitation and exploration skills can be degraded. If the exploitation capacity degrades, the number of populations generated must be increased, while if the exploration capacity deteriorates, the quality of the solution may worsen.

Other proposed algorithms that have also been shown to be effective in various areas of science and engineering but have no algorithm-specific parameters are: the sine cosine algorithm (SCA) [13] based on the sine and cosine trigonometric functions; the teaching-learning-based optimization (TLBO) algorithm [14] based on the processes of teaching and learning; the supply-demand-based optimization method (SDO) [15] based on both the demand relation of consumers and supply relation of producers; the Jaya algorithm [16] based on geometric distances and random processes; the Harris haws optimization method (HHO) [17] based on the cooperative behavior and chasing style of Harris' hawks, and Rao optimization algorithms [18]; among others.

One of the widely used techniques to improve optimization algorithms is chaos theory. Nonlinear dynamic systems that are characterized by a high sensitivity to their initial conditions are studied in chaos theory [19,20]. They can be applied to replace the PRNGs in producing the control parameters or performing local searches [21–34]. However, improving an optimization algorithm using chaotic systems instead of pseudo-random number generators (PRNGs) may be restricted to the problem under consideration or to a set of problems with similar characteristics.

Hybridization is a well-known strategy that boosts the capacity of optimization algorithms. Since a metaheuristic optimization algorithm cannot overcome all algorithms in solving any problem, hybridization can be a solution that merges the capabilities of different algorithms in one system [35–52]. Many of these algorithms require the correct setting of control parameters, and merging several of these algorithms into a single solution increases the complexity of accurate adjustment of control parameters. Furthermore, some hybridization techniques can be complicated if the management and replacement strategies of individuals in the populations are not similar. On the other hand, when chaos is applied, hybrid algorithms can provide excellent performance for a limited number of applications.

The proposed algorithms consist of hybridizations of seven of the best optimization algorithms that satisfy two requirements: (i) they must be free of algorithm-specific control parameters, and (ii) population management should allow hybridization not only at the population level but also at the individual level.

The remainder of this paper is organized as follows. Section 2 presents a brief description of the optimization algorithms used for the hybridizations. Section 3 describes the hybrid algorithms in detail, analyses of which are provided in Section 4. Finally, concluding remarks are drawn in Section 5.

2. Preliminaries

As mentioned above, among the best free control parameter algorithms are the Jaya algorithm [16], the SCA algorithm [13], the supply-demand-based optimization method [15], Rao's optimization algorithms [18], the Harris hawks optimization method (HHO) [17], and the teaching-learning-based optimization (TLBO) algorithm [14]. Among these proposals, the HHO algorithm is the most complex. It consists of two phases. During the first phase, the elements of the population are replaced without comparing the fitness of the associated solutions, which is an unwanted strategy for hybrid algorithms. In addition, the SDO algorithm, which offers impressive initial results, works with two populations preventing its integration in our hybrid proposals.

The Jaya optimization algorithm and the three new Rao's optimization algorithms (i.e., RAO1, RAO2, and RAO3) are described in Algorithm 1. The Jaya optimization algorithm has been successfully used for solving a large number of large-scale industrial problems [53–59]. The three new Rao's optimization algorithms are metaphor-less algorithms based on the best and worst solutions obtained during the optimization process and the random interactions between the candidate solutions [60–62]. In Algorithms 1–3 and 5–9, max_ITs is the number of generations; $popSize$ is the number of individuals in population Pop; $numDesignVars$ is the number of variables of the objective function F; Pop_m is the mth individual in the current population; $MinValue^k$ and $MaxValue^k$ are the low and high bounds of the kth variable of F, respectively; $BestPop$ and $WorstPop$ are the best and worst individuals of the current population, Pop, successively; $newPop_m$ is the mth new individual that can replace the current mth individual of population Pop_m; and $rand_{0..1}$ is an uniformly distributed random number in $[0, 1]$.

The SCA algorithm, presented in Algorithm 2 has been proven to be efficient in several applications [63–69].

The TLBO algorithm, described in Algorithm 3, is a two-phase algorithm; teacher phase and learner phase. It has been proven effective in solving various engineering problems [70–76].

As mentioned earlier, the use of chaotic maps can improve the behavior of some metaheuristic methods. The 2D chaotic map reported in [33] has significantly improved the convergence rate of the Jaya algorithm [33,77]. The generation of the 2D chaotic map is shown in Algorithm 4, where the initial conditions are $chA_1 = 0.2$, $chB_1 = 0.3$, $k = i$, and $dimMap = 500$. The computed values of chA_i and chB_i are in $[-1, 1]$. The chaotic Jaya algorithm (in short, CJaya) is shown in Algorithm 5, where $ch_x, x \in [1 \dots 6]$ are chaotic values randomly extracted from the 2D chaotic map. Other chaotic maps have been applied to Jaya in [32,78]. However, they do not surmount the chaotic behavior of the aforementioned 2D map.

As they present a similar structure, Algorithms 1–5 are used for designing our hybrid algorithms.

Algorithm 1 Jaya and Rao algorithms

Set max_ITs and population size (Iterator individuals: m)
Define the function cost (Iterator design variables: k)
Generate the initial population Pop
for $m = 0$ to $popSize$ **do**
  for $k = 1$ to $numDesignVars$ **do**
    $r_1 = rand_{0..1}$
    $Pop_m^k = MinValue^k + (MaxValue^k - MinValue^k) * r_1$
  end for
  Compute and store function fitness $F(Pop_m^k)$
end for
for $iterator = 1$ to max_ITs **do**
  Search for the current $BestPop$ and $WorstPop$
  for $m = 0$ to $popSize$ **do**
    Select the random individual $RandPop \neq Pop_m$ {Only for RAO2 and RAO3}
    for $k = 1$ to $numDesignVars$ **do**
      if Jaya **then**
        $r_1, r_2 = rand_{0..1}$
        $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - \left|Pop_m^k\right|\right) - r_2 \left(WorstPop^k - \left|Pop_m^k\right|\right)$
      end if
      if RAO1 **then**
        $r_1 = rand_{0..1}$
        $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - WorstPop^k\right)$
      end if
      if RAO2 **then**
        $r_1, r_2 = rand_{0..1}$
        if $Pop_m^k < RandPop^k$ **then**
          $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - WorstPop^k\right) - r_2 \left(\left|Pop_m^k\right| - \left|RandPop^k\right|\right)$
        else
          $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - WorstPop^k\right) - r_2 \left(\left|RandPop^k\right| - \left|Pop_m^k\right|\right)$
        end if
      end if
      if RAO3 **then**
        $r_1, r_2 = rand_{0..1}$
        if $Pop_m^k < RandPop^k$ **then**
          $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - \left|WorstPop^k\right|\right) - r_2 \left(\left|Pop_m^k\right| - \left|RandPop^k\right|\right)$
        else
          $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - \left|WorstPop^k\right|\right) - r_2 \left(\left|RandPop^k\right| - \left|Pop_m^k\right|\right)$
        end if
      end if
      if $newPop_m^k < MinValue^k$ **then**
        $newPop_m^k = MinValue^k$
      end if
      if $newPop_m^k > MaxValue^k$ **then**
        $newPop_m^k = MaxValue^k$
      end if
    end for
    if $F(newPop_m) < F(Pop_m)$ **then**
      $Pop_m = newPop_m$ {replace the current population}
    end if
  end for
end for
Search for the current $BestPop$

Algorithm 2 SCA optimization algorithm

Set $iniValue\_r_1 = 2$
Set max_ITs and the population size (Iterator individuals: m)
Define the function cost (Iterator design variables: k)
Generate the initial population Pop {lines 4–10 of Algorithm 1}
for $iterator = 1$ to max_ITs **do**
    Search for the current $BestPop$
    $r_1 = iniValue\_r_1 - iterator(iniValue\_r_1/max\_ITs)$
    for $m = 0$ to $popSize$ **do**
        for $k = 1$ to $numDesignVars$ **do**
            $r_2 = 2\pi rand_{0..1}$; $r_3 = 2rand_{0..1}$; $r_4 = rand_{0..1}$
            if $r_4 < 0.5$ **then**
                $newPop_m^k = Pop_m^k(r_1 \sin(r_2) \left| r_3 BestPop^k - Pop_m^k \right|)$
            else
                $newPop_m^k = Pop_m^k(r_1 \cos(r_2) \left| r_3 BestPop^k - Pop_m^k \right|)$
            end if
            Check the bounds of $newPop_m^k$ {lines 40–45 of Algorithm 1}
        end for
        if $F(newPop_m) < F(Pop_m)$ **then**
            $Pop_m = newPop_m$ {replace the current population}
        end if
    end for
end for
Search for the current $BestPop$

Algorithm 3 TLBO algorithm

Set $iniValue\_r_1 = 2$
Set max_ITs and the population size (Iterator individuals: m)
Define the function cost (Iterator design variables: k)
Generate the initial population Pop {lines 4–10 of Algorithm 1}
Set $Phase = TeacherPhase$
for $iterator = 1$ to max_ITs **do**
    Search for the current $BestPop$
    Set the teaching factor T_F (an integer random value $\in [1, 2]$)
    for $k = 1$ to $numDesignVars$ **do**
        $AveragePop^k = \left(\sum_1^m Pop^k\right) / numDesignVars$
    end for
    for $m = 0$ to $popSize$ **do**
        Select the random individual $RandPop \neq Pop_m$
        for $k = 1$ to $numDesignVars$ **do**
            if $Phase = TeacherPhase$ **then**
                $r_1 = rand_{0..1}$
                $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - T_F AveragePop^k\right)$
            end if
            if $Phase = LearnerPhase$ **then**
                $r_1 = rand_{0..1}$
                if $Pop_m^k < RandPop^k$ **then**
                    $newPop_m^k = Pop_m^k + r_1 \left(Pop_m^k - RandPop^k\right)$
                else
                    $newPop_m^k = Pop_m^k + r_1 \left(RandPop^k - Pop_m^k\right)$
                end if
            end if
            Check the bounds of $newPop_m^k$ {lines 40–45 of Algorithm 1}
        end for
        if $F(newPop_m) < F(Pop_m)$ **then**
            $Pop_m = newPop_m$ {replace the current population}
        end if
    end for
    if $Phase = TeacherPhase$ **then**
        $Phase = LearnerPhase$
    else
        $Phase = TeacherPhase$
    end if
end for
Search for the current $BestPop$

Algorithm 4 2D chaotic map

Set x_1, y_1 and $dimMap$
for $i = 1$ to $dimMap$ **do**
    $chA_{i+1} = cos(k * arccos(chB_i))$
    $chB_{i+1} = 16chA_i^5 - 20chA_i^3 + 5chA_i$
end for

Algorithm 5 Chaotic 2D Jaya algorithm

Set $iniValue\_r_1 = 2$
Set max_ITs and the population size (Iterator individuals: m)
Define the function cost (Iterator design variables: k)
Generate the initial population Pop {lines 4–10 of Algorithm 1}
for $iterator = 1$ to max_ITs **do**
    Search for the current $BestPop$
    Search for the current $WorstPop$
    Set the scaling factor S_F (integer random value $\in [1, 2]$)
    for $m = 0$ to $popSize$ **do**
        Select the random individual $RandPop \neq Pop_m$
        $r_1, r_2 = rand_{0..1}$
        $r_a = min(r_1, r_2)$
        $r_b = max(r_1, r_2)$
        for $k = 1$ to $numDesignVars$ **do**
            Extract $ch_1, ch_2, ch_3, ch_4, ch_5, ch_6$
            if $ch_1 < a$ **then**
                $newPop_m^k = ch_2 RandPop_m^k + ch_3 \left(Pop_m^k - ch_4 RandPop_m^k \right)$
                $+ch_5 \left(BestPop_m^k - ch_6 RandPop_m^k \right)$
            end if
            if $a < ch_1 < b$ **then**
                $newPop_m^k = ch_2 RandPop_m^k + ch_3 \left(Pop_m^k - ch_4 RandPop_m^k \right)$
                $+ch_5 \left(WorstPop_m^k - ch_6 RandPop_m^k \right)$
            end if
            if $ch_j > b$ **then**
                $newPop_m^k = ch_2 BestPop_m^k + ch_3 \left(RandPop_m^k - S_F BestPop_m^k \right)$
            end if
            Check the bounds of $newPop_m^k$ {lines 40–45 of Algorithm 1}
        end for
        if $F(newPop_m) < F(Pop_m)$ **then**
            $Pop_m = newPop_m$ {replace the current population}
        end if
    end for
end for
Search for the current $BestPop$

3. Hybrid Algorithms

The proposed hybrid algorithms are designed using the seven algorithms described in Section 2. These algorithms have been selected thanks to their performance in solving constrained and unconstrained functions, but also because they share a similar structure that allows the implementation of different hybridization strategies.

Algorithm 6 shows the skeleton of the proposed hybrid algorithms, which includes all common and uncommon tasks without any updating procedure of the current population. Since the TLBO algorithm is a two-phase algorithm, the proposed hybrid algorithms apply these two phases consecutively to each individual. In contrast to the other algorithm where a single-phase is executed, a control parameter *Phase* is applied to process twice the same individual when the TLBO algorithm is used (see lines 24–29 of Algorithm 6). The algorithm used to obtain a new individual is determined by *AlgSelected* (see line 17 of Algorithm 6). In Algorithms 6–9, *AlgSelected* determines the algorithm accountable for producing a new individual.

Given that only algorithms that are free of control parameters have been considered, proposals that require the inclusion of control parameters have been discarded. Following these guidelines, we have designed three hybrid algorithms, an analysis of which is provided in Section 4. The first proposed hybrid algorithm, shown in Algorithm 7, processes the entire population in each iteration using the same algorithm, and is referred to as the HYBPOP algorithm. This is the most straightforward hybridization technique where the requirement to follow the structure given by Algorithm 6 is not

mandatory on all algorithms. In Algorithms 7–9, $NumOfAlgorithms$ is the number of algorithms free of control parameters involved in the hybrid proposals.

Algorithm 6 Skeleton of hybrid algorithms

Set max_ITs and the population size (Iterator individuals: m)
Define the function cost (Iterator design variables: k)
Set $iniValue\_r_1 = 2$; $Phase = TeacherPhase$; $RepeatTLBO = false$
Generate the initial population Pop {lines 4–10 of Algorithm 1}
for $iterator = 1$ to max_ITs **do**
    Search for the current $BestPop$ and $WorstPop$
    Set the scaling factor S_F and teaching factor T_F (an integer random value $\in [1,2]$)
    for $k = 1$ to $numDesignVars$ **do**
        $AveragePop^k = \left(\sum_1^m Pop^k\right) / numDesignVars$
    end for
    $r_1 = iniValue\_r_1 - iterator(iniValue\_r_1 / max\_ITs)$
    for $m = 0$ to $popSize$ **do**
        Select random individual $RandPop \neq Pop_m$
        $r_2, r_3 = rand_{0..1}$
        $r_a = min(r_2, r_3)$; $r_b = max(r_2, r_3)$
        for $k = 1$ to $numDesignVars$ **do**
            $\Rightarrow$ ($AlgSelected$) Compute $newPop_m^k$ using $AlgSelected$ (one from Algorithms 1–5)
            Check the bounds of $newPop_m^k$ {lines 40–45 of Algorithm 1}
        end for
        if $F(newPop_m) < F(Pop_m)$ **then**
            $Pop_m = newPop_m$ {Replace the current population}
        end if
        if $TLBO$ **then**
            if $Phase = TeacherPhase$ **then**
                $Phase = LearnerPhase$; $RepeatTLBO = true$;$m = m - 1$
            else
                $Phase = TeacherPhase$; $RepeatTLBO = false$
            end if
        end if
    end for
end for
Search for the current $BestPop$

Algorithm 7 HYBPOP: Hybrid algorithm based on population

$NumOfAlgorithms = 7$
$ALGS[NumOfAlgorithms] = \{$Jaya, Chaotic Jaya, SCA, RAO1, RAO2, RAO3, TLBO$\}$
$Selection = 0$
for $iterator = 1$ to max_ITs **do**
    $AlgSelected = ALGS[Selection]$
    for $m = 0$ to $popSize$ **do**
        for $k = 1$ to $numDesignVars$ **do**
            Compute $newPop_m^k$ using $AlgSelected$
        end for
    end for
    $Selection = Selection + 1$
    if $Selection \geq NumOfAlgorithms$ **then**
        $Selection = 0$
    end if
end for
Search for the current $BestPop$

The second algorithm, named HYBSUBPOP, is described through Algorithm 8. It logically splits the population into sub-populations. During the optimization process, each sub-population will be processed by one of the seven algorithms mentioned previously.

Algorithm 8 HYBSUBPOP: Hybrid algorithm based on sub-populations

```
NumOfAlgorithms = 7
Split popSize into NumOfAlgorithms sub-populations
ALGS[NumOfAlgorithms] = {Jaya, Chaotic Jaya, SCA, RAO1, RAO2, RAO3, TLBO}
for iterator = 1 to max_ITs do
    for m = 0 to popSize do
        subPopID = sub-population index of individual m.
        AlgSelected = ALGS[subPopID]
        for k = 1 to numDesignVars do
            Compute newPop_m^k using AlgSelected
        end for
    end for
end for
Search for the current BestPop
```

Algorithm 9 shows the third proposed hybrid algorithm, dubbed HYBIND, in which a different algorithm in each iteration handles each individual of the population.

Algorithm 9 HYBIND: Hybrid algorithm based on individuals

```
NumOfAlgorithms = 7
ALGS[NumOfAlgorithms] = {Jaya, Chaotic Jaya, SCA, RAO1, RAO2, RAO3, TLBO}
Selection = 0
for iterator = 1 to max_ITs do
    Selection = iterator%7
    for m = 0 to popSize do
        AlgSelected = ALGS[Selection]
        for k = 1 to numDesignVars do
            Compute newPop_m^k using AlgSelected
        end for
        Selection = Selection + 1
        if Selection ≥ NumOfAlgorithms then
            Selection = 0
        end if
    end for
end for
Search for the current BestPop
```

It is worth noting that the aim of the proposed hybrid algorithms is not to improve the convergence ratio of the used algorithms separately, nor to perform optimally for a particular problem. It is to show outstanding performance for a large number of problems without adjusting any control parameters of the considered algorithms.

4. Numerical Experiments

In this section, the performance of the proposed hybrid algorithms is analyzed through solving 28 well-known unconstrained functions (see Table 1), the definitions of which can be seen in [77]. The proposed algorithms were implemented in the C language, using the GCC v.4.4.7 [79], and an Intel Xeon E5-2620 v2 processor at 2.1 GHz. The hybrid proposals, along with the original algorithms, have been implemented and tested using C language. The C implementations of the original algorithms used are not available through the Internet. However, their Java/Matlab implementations are commonly available.

Table 1. Benchmark functions. Names and parameters.

Id.	Name	Num. vars (D)	Domain (Min, Max)	Id.	Name	Num. vars (D)	Domain (Min, Max)
F1	Sphere	30	$-100, 100$	F15	Bohachevsky_1	2	$-100, 100$
F2	SumSquares	30	$-10, 10$	F16	Booth	2	$-10, 10$
F3	Beale	2	$-4.5, 4.5$	F17	Michalewicz_2	2	$0, \pi$
F4	Easom	2	$-100, 100$	F18	Michalewicz_5	5	$0, \pi$
F5	Matyas	2	$-10, 10$	F19	Bohachevsky_2	2	$-100, 100$
F6	Colville	4	$-10, 10$	F20	Bohachevsky_3	2	$-100, 100$
F7	Trid 6	6	$-D^2, D^2$	F21	GoldStein-Price	2	$-2, 2$
F8	Trid 10	10	$-D^2, D^2$	F22	Perm	4	$-D, D$
F9	Zakharov	10	$-5, 10$	F23	Hartman_3	3	$0, 1$
F10	Schwefel_1.2	30	$-100, 100$	F24	Ackley	30	$-32, 32$
F11	Rosenbrock	30	$-30, 30$	F25	Penalized_2	30	$-50, 50$
F12	Dixon-Price	5	$-10, 10$	F26	Langermann_2	2	$0, 10$
F13	Foxholes	2	$-2^{16}, 2^{16}$	F27	Langermann_5	5	$0, 10$
F14	Branin	2	$x_1 : -5, 10$ $x_2 : 0, 15$	F28	Fletcher-Powell_5	5	$x_i, \alpha_i : -\pi, \pi$ $a_{ij}, b_{ij} : -100, 100$

The data collected from the experimental analysis are as follows:

- **NoR-AI**: the total number of replacements for any individual.
- **NoR-BI**: the total number of replacements for the current best individual.
- **NoR-BwT**: the total number of replacements for the current best individual with an error of less than 0.001.
- **LtI-AI**: the last iteration (*iterator*) in which a replacement of any individual occurs.
- **LtI-BI**: the last iteration (*iterator*) in which a replacement of the best individual occurs.

Three of the five analyzed data (NoR-) indicate the number of times the current individual (Pop_m) is replaced by a new individual ($newPop_m$), which provides a better fitness function (see line 21 of Algoritm 6), while the remaining two (LtI-) refer to the last generation (iterator) in which at least one individual has been replaced.

All data given below have been obtained under 50 runs, $50,000$ iterations ($max\_ITs = 50,000$) and two population sizes ($popSize = 140$ and 210). The maximum values of the analyzed data are listed in Table 2.

Table 2. Maximum values of the analyzed data.

Population Size	70	140	210
NoR-AI	3,500,000	7,000,000	10,500,000
NoR-BI	3,500,000	7,000,000	10,500,000
NoR-BwT	3,500,000	7,000,000	10,500,000
LtI-AI	49,999	49,999	49,999
LtI-BI	49,999	49,999	49,999

Tables 3–5 show the data of all the considered algorithms independently, i.e., without hybridization. As expected, the behavior of the different algorithms does not follow a familiar pattern. In addition, it depends on the objective function. Regarding a global convergence analysis, both TLBO and CJaya behave better but with a higher order of complexity (see [77,80]). Moreover, it is noted that when using

TLBO, two new individuals are generated in each iteration; one in the teacher phase and the other one in the learner phase. The values in brackets in Tables 3–5 refer to the standard deviation of the data under 50 runs. Note that heuristic optimization algorithms are partially based on randomness, which leads to high values of standard deviation. The average standard deviations are approximately equal to 16%, 22%, 15%, 30%, 23%, 23%, and 22% for Jaya, Chaotic Jaya, SCA, RAO1, RAO2, RAO3, and TLBO, respectively.

An important aspect, not shown in Tables 3–5, is whether the solution obtained by each algorithm is acceptable or not. In particular, the original algorithms fail to obtain a solution tolerance of less than 0.001 for 3, 8, 2, 4, 7, 5, and 2 functions for Jaya, CJaya, SCA, RAO1, RAO2, RAO3, and TLBO, respectively. Therefore, considering only original algorithms, there is no algorithm whose behavior is always the best, which justifies the development of a generalist hybrid system that can solve a large number of benchmark functions and engineering problems.

Comparing the quality of the solutions obtained from the proposed hybrid algorithms, it can be concluded that the HYBSUBPOP algorithm is the worst one because the same thoroughbred algorithm is always applied to the same sub-population, which degrades the algorithm's performance for a small population. Contrary to HYBSUBPOP, the HYBPOP and HYBIND algorithms apply the selected algorithms to all individuals, which leads to better-exploiting hybridizations. The HYBSUPOP algorithm fails to obtain a solution tolerance of less than 0.001 in 3 functions (F11, F23, and F27) and the HYBPOP and HYBIND algorithms fail in only one function (F27 and F11, respectively). If the population size is increased to 210, the HYBIND algorithm succeeds with all functions, thus the HYBIND algorithm has a slightly better performance in comparison to HYBPOP.

Local exploration has improved both in the HYBPOP method and especially in the HYBIND method, as stated above. Figures 1 and 2 show the convergence curves of both all the individual methods and the three hybrid methods proposed for the first 1000 and 100 iterations, respectively, for functions F1, F8, F11, and F18. Each point in both figures is the average of the data obtained from 10 runs. As shown in these figures, the curves of the three hybrid methods are similar to the curves of the best single algorithms for each function. Therefore, global exploitation, while not improving all methods, behaves similarly to the best single methods for each function. It should be noted that the hybrid methods behave similarly to the best individual methods for each function, which are not always the same.

Table 6 sorts the algorithms according to the number of iterations required to obtain an error of less than 0.001, if an algorithm is missing in a row an acceptable solution is not reached. As seen from this table, no algorithm outperforms all other algorithms. Moreover, a computational cost analysis would be necessary to classify them correctly. Table 7 exhibits the computational cost of different algorithms. It reveals from this table that the hybrid algorithms are mid-ranked in terms of computational cost, and HYBIND is computationally less expensive than HYBPOP.

An analysis of the contribution of each algorithm in the HYBPOP and HYBIND algorithms is exhibited in Tables 8–10. Table 8 indicates the number of times that an individual has been replaced in each algorithm. The replacement is accepted when the new individual improves the fitness of the current solution. As seen from Table 8, the HYBIND algorithm performs more replacements of individuals. In addition, the numbers of replacements per individual for the contributing algorithms are nearly equal, except for the RAO1 algorithm, where the contribution to replacements is limited. The standard deviations of each data (from 50 runs) are being put in brackets. We found that, on average, the standard deviations for HYBPOP and HYBRID algorithms are both equal to 14%.

Table 9 shows the last iteration in which each optimization algorithm replaces an individual in the population, i.e., when it no longer brings improvement to the hybrid algorithm. As can be seen from Table 8, the optimization algorithms, except the RAO1 algorithm, work efficiently in the hybrid algorithms. It is also revealed that the considered algorithms contribute to more generations in the HYBIND algorithm. The mean value of the standard deviation rises to 28% and 23% for HYBPOP and HYBIND, respectively, due to the randomness behavior and lower LtI-AI costs.

Table 3. Analysis of Jaya and chaotic Jaya on function F1-F28 with a population size of 140.

	Jaya					Chaotic Jaya				
	NoR -AI	**NoR -BI**	**NoR -BwT**	**LtI -AI**	**LtI -BI**	**NoR -AI**	**NoR -BI**	**NoR -BwT**	**LtI -AI**	**LtI -BI**
F1	293,516	3676	3434	49,999	49,971	75,435	1580	1525	1168	1157
	(1323)	(37)	(34)	(0)	(27)	(421)	(58)	(55)	(5)	(5)
F2	294,070	3653	3435	49,999	49,990	75,324	1581	1526	1163	1151
	(1444)	(75)	(68)	(0)	(8)	(449)	(51)	(50)	(8)	(8)
F3	8957	70	60	862	778	2394	20	9	49,719	28,810
	(107)	(7)	(7)	(164)	(163)	(52)	(4)	(2)	(168)	(8641)
F4	5045	43	24	451	412	2453	26	8	49,773	27,786
	(69)	(4)	(4)	(77)	(74)	(41)	(5)	(2)	(214)	(12,758)
F5	96,251	739	729	8351	8236	1930	14	8	41	8
	(253)	(32)	(34)	(65)	(65)	(528)	(11)	(7)	(12)	(6)
F6	17,568	151	112	25,293	22,919	3264	54	0	49,843	41,426
	(129)	(15)	(11)	(1564)	(1752)	(358)	(19)	(0)	(122)	(11,336)
F7	11,683	115	56	49,044	10,094	3856	60	0	49,722	37,363
	(80)	(8)	(4)	(1099)	(8036)	(275)	(10)	(0)	(339)	(7744)
F8	18,075	197	79	49,358	28,600	6030	114	0	49,722	35,566
	(658)	(28)	(6)	(561)	(9924)	(743)	(23)	(0)	(303)	(10,334)
F9	249,027	2288	2206	49,999	49,968	27,271	443	421	552	446
	(939)	(38)	(40)	(0)	(18)	(12,031)	(289)	(264)	(281)	(210)
F10	7522	70	0	49,993	48,950	76,978	1465	1411	1597	1552
	(380)	(12)	(0)	(6)	(1026)	(1835)	(109)	(109)	(37)	(116)
F11	59,193	956	375	49,992	49,660	9367	197	0	49,925	43,912
	(7991)	(239)	(174)	(11)	(773)	(1936)	(48)	(0)	(47)	(7236)
F12	7648	74	27	41,236	25,124	3519	56	0	49,765	34,756
	(891)	(11)	(10)	(14,771)	(18,788)	(70)	(5)	(0)	(320)	(11,038)
F13	2658	23	0	49,779	30,524	3220	37	0	49,666	38,222
	(73)	(3)	(0)	(272)	(10,879)	(553)	(10)	(0)	(292)	(9294)
F14	3261	29	0	21719	17234	2023	21	0	49,713	32,070
	(925)	(7)	(0)	(18,396)	(16,324)	(111)	(5)	(0)	(253)	(11,911)
F15	6084	42	22	275	234	1639	21	8	30	11
	(62)	(4)	(3)	(9)	(9)	(258)	(12)	(3)	(2)	(6)
F16	2827	24	8	49,678	38,126	2413	27	9	49,709	34,212
	(71)	(4)	(1)	(261)	(10,321)	(41)	(5)	(3)	(227)	(12,119)
F17	4945	36	1	4098	174	2134	18	1	49,739	34,560
	(52)	(6)	(0)	(1714)	(12)	(47)	(4)	(0)	(333)	(12,597)
F18	9958	96	0	48,344	8329	2162	20	0	49,648	36,491
	(134)	(10)	(0)	(723)	(1549)	(60)	(3)	(0)	(320)	(10,256)
F19	6247	46	25	329	273	1708	19	6	32	10
	(53)	(6)	(6)	(20)	(18)	(315)	(13)	(2)	(4)	(5)
F20	6258	48	24	657	444	1597	16	4	31	7
	(70)	(6)	(4)	(251)	(27)	(217)	(9)	(1)	(2)	(2)
F21	2599	18	4	49,818	37,023	2543	28	14	49,651	27,864
	(52)	(4)	(1)	(141)	(10,059)	(36)	(5)	(2)	(241)	(14,620)
F22	1841	15	0	48,784	18,724	1907	17	0	49,850	24,366
	(176)	(3)	(0)	(1361)	(10,505)	(92)	(3)	(0)	(123)	(11,825)
F23	5976	51	0	394	180	2170	19	0	49,662	23,570
	(84)	(6)	(0)	(81)	(9)	(35)	(4)	(0)	(323)	(12,705)
F24	41,575	415	205	9044	4874	6509	117	55	124	95
	(269)	(23)	(12)	(5088)	(2387)	(158)	(25)	(19)	(11)	(17)
F25	58,079	840	621	8652	8613	5559	75	0	49,810	40,191
	(593)	(25)	(20)	(111)	(111)	(852)	(20)	(0)	(255)	(7868)
F26	5063	43	0	13,854	11,630	2389	22	0	49,848	32,604
	(110)	(5)	(0)	(8815)	(5926)	(200)	(7)	(0)	(116)	(13,247)
F27	9071	79	23	25,536	9665	2127	21	0	49,698	31,794
	(452)	(9)	(25)	(18,586)	(3926)	(140)	(4)	(0)	(284)	(11,266)
F28	2119	15	0	49,427	24,846	2156	21	0	49,635	33,067
	(159)	(2)	(0)	(751)	(11,088)	(124)	(6)	(0)	(353)	(12,178)

Table 4. Analysis of sine cosine algorithm (SCA) and Rao's optimization Algorithm 1 (RAO1) on function F1-F28 with a population size of 140.

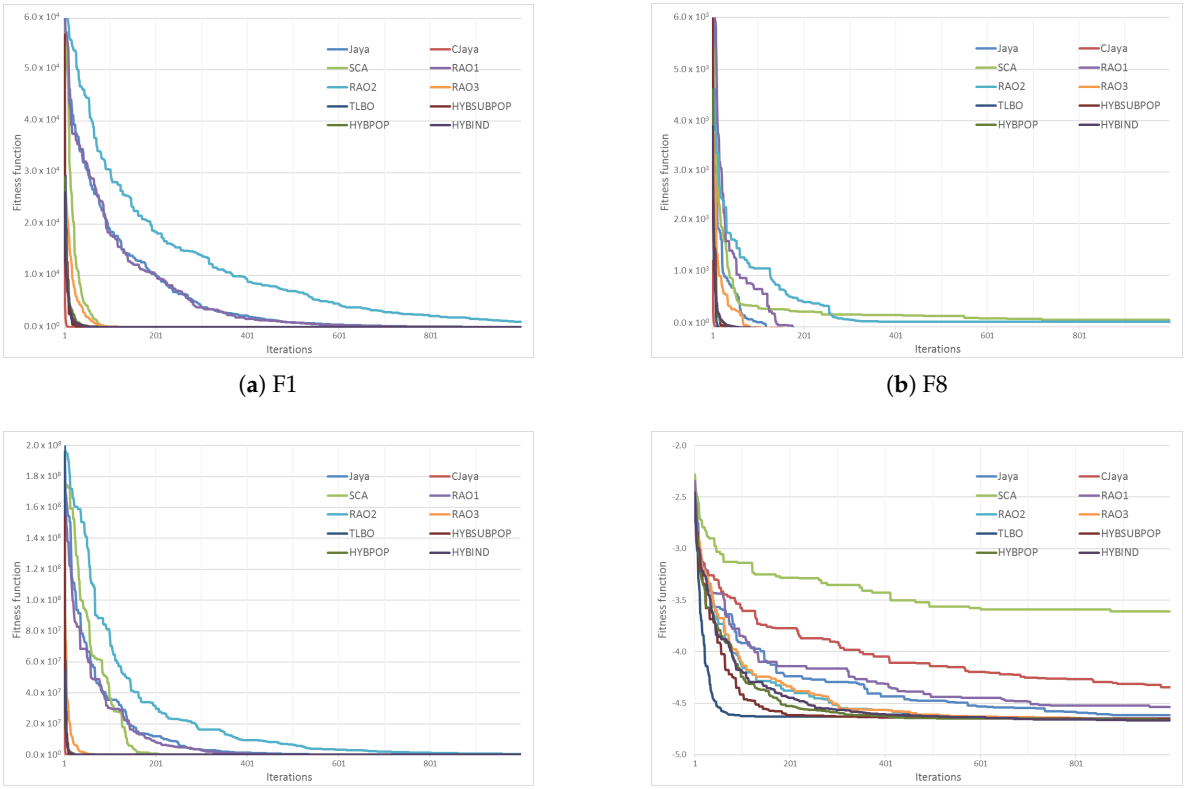
	SCA					RAO1				
	NoR -AI	**NoR -BI**	**NoR -BwT**	**LtI -AI**	**LtI -BI**	**NoR -AI**	**NoR -BI**	**NoR -BwT**	**LtI -AI**	**LtI -BI**
F1	209,388 (323)	1397 (24)	1339 (23)	12,387 (440)	4261 (181)	282,204 (671)	4109 (35)	3847 (30)	49,999 (0)	49,985 (15)
F2	208,824 (335)	1397 (16)	1344 (15)	12,129 (232)	4222 (99)	282,675 (1315)	4105 (47)	3867 (43)	49,999 (0)	49,990 (7)
F3	3263 (59)	24 (5)	11 (2)	49,999 (0)	49,908 (252)	10,490 (104)	71 (5)	59 (5)	1037 (122)	937 (124)
F4	3522 (71)	30 (5)	10 (3)	49,999 (0)	49,983 (27)	5921 (61)	41 (5)	23 (3)	819 (431)	588 (79)
F5	81,841 (285)	631 (25)	622 (24)	4113 (89)	2540 (177)	102,605 (355)	739 (23)	728 (23)	6077 (21)	5996 (23)
F6	6613 (158)	42 (6)	0 (0)	49,999 (0)	49,997 (2)	20,964 (211)	154 (8)	111 (6)	45,874 (1909)	41,787 (2043)
F7	7416 (89)	60 (7)	0 (0)	49,999 (0)	49,999 (0)	14,501 (132)	114 (9)	57 (6)	48,669 (1124)	19,652 (15,645)
F8	11,411 (197)	90 (12)	0 (0)	49,999 (0)	49,998 (0)	21,868 (505)	190 (14)	77 (4)	49,782 (242)	23,351 (13,762)
F9	137,414 (296)	1134 (26)	1098 (24)	13,643 (141)	7062 (415)	394,478 (748)	3436 (58)	3350 (58)	49,992 (15)	49,865 (102)
F10	131,216 (572)	1416 (18)	1352 (18)	20,314 (684)	13,659 (992)	50,688 (792)	573 (13)	299 (16)	49,999 (0)	49,912 (58)
F11	22,671 (509)	173 (11)	0 (0)	49,999 (0)	49,998 (1)	49,031 (5791)	730 (88)	116 (56)	49,986 (15)	49,824 (147)
F12	7116 (232)	59 (8)	8 (3)	49,999 (0)	49,996 (5)	18,005 (6110)	144 (51)	97 (49)	15,151 (12,276)	9236 (6218)
F13	4083 (70)	37 (7)	0 (0)	49,999 (0)	49,996 (3)	1174 (2874)	9 (21)	0 (0)	9811 (24,036)	8799 (21,554)
F14	3161 (53)	22 (6)	0 (0)	49,999 (0)	49,985 (36)	2342 (185)	18 (3)	0 (0)	45,878 (4245)	28,668 (18,831)
F15	5976 (33)	43 (6)	22 (5)	200 (19)	66 (6)	6468 (87)	46 (5)	24 (5)	263 (28)	217 (9)
F16	3314 (57)	27 (3)	10 (2)	49,999 (0)	49,998 (1)	10,022 (135)	70 (11)	53 (9)	352 (9)	298 (4)
F17	3030 (30)	24 (2)	0 (0)	49,999 (0)	49,994 (9)	5717 (80)	39 (5)	1 (1)	4279 (749)	221 (12)
F18	5461 (100)	45 (4)	0 (0)	4,9999 (0)	49,998 (2)	12,198 (76)	100 (11)	0 (0)	47,566 (3038)	7285 (1664)
F19	6201 (71)	41 (7)	22 (2)	259 (30)	68 (4)	6648 (89)	45 (5)	25 (3)	318 (12)	268 (7)
F20	5751 (66)	42 (3)	23 (4)	493 (46)	101 (10)	6677 (88)	42 (5)	25 (6)	10,328 (19,703)	389 (26)
F21	3330 (45)	26 (4)	12 (3)	49,999 (0)	48,236 (4088)	6390 (126)	48 (5)	31 (3)	41,485 (6441)	22,033 (13,841)
F22	2986 (76)	25 (6)	0 (0)	49,999 (0)	49,975 (37)	1931 (162)	13 (2)	0 (0)	49,470 (805)	24,078 (13,173)
F23	3699 (97)	29 (6)	0 (0)	49,999 (0)	49,951 (109)	6505 (2170)	53 (17)	0 (0)	670 (379)	215 (72)
F24	22,109 (3516)	128 (7)	62 (4)	26,470 (9520)	4247 (6275)	45,468 (342)	441 (15)	217 (15)	8818 (7583)	4045 (500)
F25	23,889 (117)	206 (12)	0 (0)	49,999 (0)	49,998 (1)	59,024 (431)	843 (18)	618 (11)	10,895 (61)	10,755 (61)
F26	3205 (42)	21 (4)	0 (0)	49,999 (0)	49,958 (48)	5850 (130)	41 (6)	0 (0)	18,757 (11,293)	17,964 (11,291)
F27	5376 (621)	47 (8)	0 (1)	49,999 (0)	49,997 (2)	6280 (4993)	47 (36)	10 (23)	32,659 (10,816)	14,141 (8996)
F28	5969 (241)	42 (3)	0 (0)	49,999 (0)	49,997 (2)	4077 (1375)	33 (14)	0 (0)	45,602 (4353)	30,802 (13,495)

Table 5. Analysis of RAO2, RAO3, and teaching-learning-based optimization (TLBO) on function F1-F28 with a population size of 140.

	RAO2					RAO3					TLBO				
	NoR-AI	**NoR-BI**	**NoR-BwT**	**LtI-AI**	**LtI-BI**	**NoR-AI**	**NoR-BI**	**NoR-BwT**	**LtI-AI**	**LtI-BI**	**NoR-AI**	**NoR-BI**	**NoR-BwT**	**LtI-AI**	**LtI-BI**
F1	158,207	1833	1582	49,999	49,975	402,040	6191	5990	12,845	12,588	317,548	6636	6432	1816	1806
	(321)	(35)	(25)	(0)	(20)	(1764)	(59)	(52)	(171)	(152)	(2268)	(62)	(62)	(5)	(4)
F2	158,651	1838	1600	49,999	49,964	400,507	6179	5990	12,748	12,545	314,216	6582	6388	1802	1792
	(522)	(36)	(30)	(0)	(30)	(1638)	(77)	(71)	(140)	(122)	(1213)	(82)	(79)	(5)	(4)
F3	9855	74	59	696	610	9933	74	60	665	583	10,098	72	60	296	166
	(93)	(11)	(9)	(40)	(45)	(111)	(6)	(6)	(32)	(32)	(209)	(7)	(7)	(81)	(8)
F4	5435	44	25	2038	2015	5509	44	25	1088	1063	5455	48	25	127	83
	(79)	(9)	(6)	(899)	(897)	(75)	(6)	(5)	(614)	(616)	(124)	(5)	(4)	(19)	(4)
F5	107,111	736	727	8885	8769	115,700	756	744	12,296	12,161	151,056	764	751	1261	1229
	(218)	(10)	(10)	(81)	(77)	(411)	(26)	(24)	(180)	(190)	(1162)	(26)	(25)	(5)	(6)
F6	16,646	129	93	49,995	49,669	16,154	127	86	49,984	49,810	46,797	185	136	5463	3573
	(542)	(12)	(8)	(3)	(402)	(615)	(11)	(10)	(20)	(159)	(3371)	(17)	(7)	(677)	(250)
F7	11,737	108	56	48,902	7854	11,421	115	61	48,508	13,021	13,459	139	70	49,638	5639
	(91)	(5)	(5)	(865)	(5216)	(172)	(8)	(3)	(2285)	(9489)	(501)	(12)	(9)	(279)	(1539)
F8	17,869	218	63	49,543	19,788	17,255	188	76	48,896	25,615	202,435	3127	1418	49,440	24,482
	(1951)	(45)	(32)	(413)	(5571)	(574)	(15)	(9)	(1466)	(9085)	(46,668)	(779)	(176)	(444)	(14,497)
F9	158,199	1411	1324	49,999	49,965	209,334	2857	2785	17,784	17,582	277,040	3261	3180	2985	2958
	(509)	(41)	(38)	(0)	(34)	(765)	(50)	(50)	(301)	(277)	(1370)	(53)	(55)	(11)	(12)
F10	3889	31	0	49,943	46,966	58,363	716	536	49,998	49,889	409,579	5276	5097	8337	8270
	(267)	(4)	(0)	(52)	(2679)	(4038)	(60)	(62)	(1)	(103)	(1597)	(55)	(60)	(65)	(71)
F11	34,536	524	8	49,940	49,106	47,480	905	7	49,984	49,761	4,325,394	65,328	26,532	42,857	38,436
	(5740)	(107)	(7)	(79)	(790)	(2960)	(71)	(8)	(17)	(197)	(204,520)	(3160)	(3689)	(3655)	(2036)
F12	19,104	187	136	1472	1393	12,595	123	57	6960	3570	25,139	234	172	393	299
	(215)	(13)	(12)	(917)	(85)	(5136)	(54)	(66)	(2906)	(619)	(3571)	(11)	(17)	(190)	(161)
F13	640	9	0	49,680	21,886	1285	14	0	49,649	30,857	5251	49	0	48,857	1302
	(44)	(2)	(0)	(301)	(9803)	(794)	(5)	(0)	(532)	(13,644)	(368)	(10)	(0)	(1119)	(467)
F14	4472	30	0	4716	1414	4423	32	0	6115	3465	4862	33	0	2518	64
	(752)	(7)	(0)	(684)	(683)	(767)	(7)	(0)	(3259)	(3260)	(125)	(5)	(0)	(225)	(11)
F15	6939	43	22	667	348	6474	43	23	319	275	8420	48	26	63	51
	(126)	(5)	(3)	(166)	(31)	(72)	(4)	(2)	(17)	(17)	(320)	(5)	(3)	(1)	(1)
F16	9812	69	53	534	464	2904	22	7	49,838	32,614	11,723	69	54	103	86
	(85)	(8)	(6)	(29)	(20)	(42)	(5)	(2)	(177)	(13,514)	(260)	(5)	(5)	(2)	(1)
F17	5089	40	1	2641	149	5103	40	1	2187	144	5888	35	1	2137	56
	(55)	(5)	(0)	(726)	(7)	(70)	(4)	(1)	(425)	(9)	(330)	(5)	(0)	(713)	(7)
F18	9466	95	0	48,411	1938	9493	93	0	49151	2102	14,358	122	0	49,208	1466
	(106)	(8)	(0)	(1305)	(786)	(136)	(6)	(0)	(579)	(570)	(3227)	(23)	(0)	(668)	(866)

Table 5. *Cont.*

	RAO2					RAO3					TLBO				
	NoR-AI	**NoR-BI**	**NoR-BwT**	**LtI-AI**	**LtI-BI**	**NoR-AI**	**NoR-BI**	**NoR-BwT**	**LtI-AI**	**LtI-BI**	**NoR-AI**	**NoR-BI**	**NoR-BwT**	**LtI-AI**	**LtI-BI**
F19	7104	49	27	833	547	6677	46	24	561	509	7812	43	21	73	58
	(123)	(6)	(5)	(77)	(96)	(99)	(4)	(4)	(225)	(220)	(327)	(2)	(3)	(3)	(2)
F20	6847	42	23	3782	1677	5730	45	26	1115	1044	8677	47	25	103	75
	(122)	(4)	(2)	(894)	(867)	(77)	(7)	(5)	(568)	(569)	(563)	(6)	(5)	(6)	(5)
F21	6889	45	28	49,200	10,483	2627	21	4	49,803	33,314	6407	46	30	27,592	9567
	(86)	(4)	(2)	(1009)	(4233)	(82)	(4)	(1)	(141)	(10,046)	(348)	(8)	(6)	(16,851)	(11,094)
F22	2050	15	0	47,791	22,625	2381	23	0	49,441	35,495	27,875	218	55	49,997	47,112
	(134)	(3)	(0)	(3154)	(14,711)	(502)	(8)	(0)	(729)	(14,282)	(2436)	(147)	(1)	(2)	(3039)
F23	6113	49	0	241	151	6095	52	0	249	147	7185	57	0	149	57
	(58)	(4)	(0)	(19)	(3)	(67)	(5)	(0)	(26)	(4)	(323)	(6)	(0)	(27)	(2)
F24	38,389	371	182	32,501	15,916	27,517	355	177	1532	841	25,248	402	201	237	167
	(18515)	(12)	(91)	(12,156)	(12,065)	(262)	(19)	(10)	(987)	(36)	(557)	(16)	(15)	(17)	(8)
F25	52,014	758	554	4811	4799	49,682	741	545	1584	1570	140,740	2360	837	21,162	7098
	(331)	(34)	(24)	(210)	(209)	(265)	(33)	(25)	(37)	(35)	(43,787)	(744)	(1381)	(22,046)	(3282)
F26	5153	39	0	1061	883	5263	39	0	6302	1436	6689	38	0	15,868	758
	(121)	(5)	(0)	(882)	(870)	(242)	(5)	(0)	(1120)	(970)	(956)	(7)	(0)	(4085)	(373)
F27	9039	93	16	14,656	4382	9148	92	23	19,748	2674	12,647	222	124	21,151	5807
	(372)	(5)	(27)	(543)	(5900)	(289)	(5)	(28)	(23,981)	(2360)	(1424)	(63)	(42)	(11,703)	(3567)
F28	3993	33	0	43,661	35,225	2318	18	0	49,411	26,332	38,285	238	162	19,500	897
	(525)	(8)	(0)	(10,700)	(13,441)	(291)	(5)	(0)	(561)	(13,590)	(4443)	(13)	(19)	(19,909)	(345)



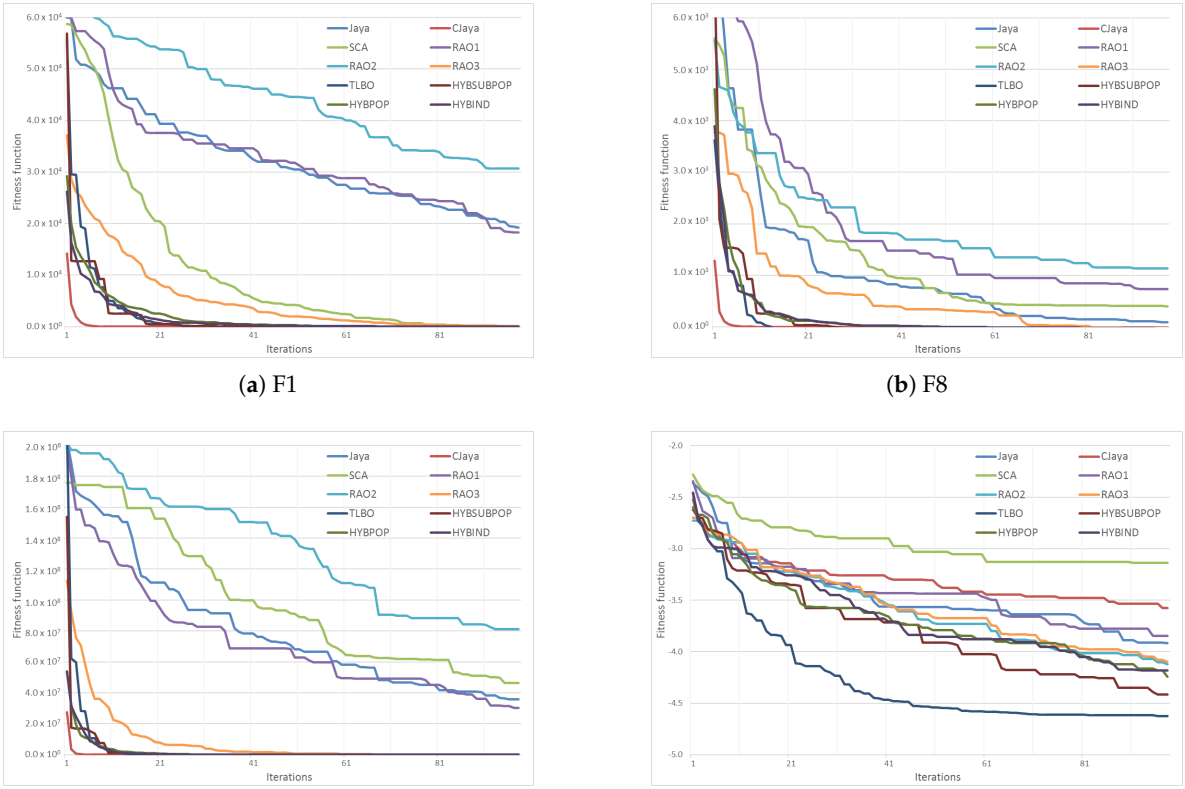
(**a**) F1 (**b**) F8

(**c**) F11 (**d**) F18

Figure 1. Convergence curves. The population size is set as 140 and the number of iterations to 1000.

Table 6. Ranking of the algorithms according to the number of iterations required to achieve an error of less than 0.001. The population size is set as 140.

	1	2	3	4	5	6	7	8	9	10
F1	CJaya	TLBO	HYBPOP	SCA	HYBIND	RAO3	HYBSUBPOP	RAO1	Jaya	RAO2
F2	CJaya	TLBO	HYBPOP	SCA	HYBIND	HYBSUBPOP	RAO3	RAO1	Jaya	RAO2
F3	TLBO	HYBPOP	CJaya	HYBSUBPOP	HYBIND	RAO3	SCA	RAO2	Jaya	RAO1
F4	CJaya	HYBPOP	TLBO	HYBSUBPOP	HYBIND	SCA	Jaya	RAO1	RAO2	RAO3
F5	HYBPOP	SCA	HYBIND	HYBSUBPOP	TLBO	RAO3	Jaya	RAO1	RAO2	CJaya
F6	TLBO	HYBPOP	Jaya	HYBIND	HYBSUBPOP	RAO3	RAO1	RAO2	SCA	CJaya
F7	TLBO	HYBPOP	RAO3	Jaya	RAO2	RAO1	HYBIND	HYBSUBPOP	SCA	CJaya
F8	HYBPOP	Jaya	RAO3	TLBO	RAO1	HYBIND	HYBSUBPOP	SCA		
F9	CJaya	HYBPOP	TLBO	HYBIND	HYBSUBPOP	SCA	RAO3	RAO1	Jaya	RAO2
F10	CJaya	HYBPOP	TLBO	HYBIND	HYBSUBPOP	SCA	RAO3	RAO1		
F11	HYBPOP	Jaya	RAO1	TLBO	RAO3	RAO2				
F12	TLBO	RAO2	RAO1	HYBIND	Jaya	HYBSUBPOP	HYBPOP	SCA		
F13	HYBPOP	SCA	HYBIND	HYBSUBPOP	TLBO	Jaya				
F14	TLBO	CJaya	HYBSUBPOP	RAO2	HYBIND	HYBPOP	RAO3	Jaya	SCA	RAO1
F15	CJaya	HYBPOP	SCA	HYBSUBPOP	HYBIND	TLBO	Jaya	RAO1	RAO3	RAO2
F16	TLBO	HYBPOP	RAO1	CJaya	HYBSUBPOP	HYBIND	RAO2	RAO3	Jaya	SCA
F17	TLBO	HYBPOP	HYBIND	Jaya	HYBSUBPOP	RAO2	RAO3	RAO1	CJaya	SCA
F18	HYBIND	HYBSUBPOP	Jaya	RAO1	SCA	HYBPOP				
F19	SCA	HYBPOP	HYBIND	TLBO	Jaya	RAO1	RAO3	RAO2	CJaya	HYBSUBPOP
F20	SCA	HYBPOP	HYBSUBPOP	HYBIND	TLBO	Jaya	RAO1	RAO3	RAO2	CJaya
F21	CJaya	TLBO	HYBPOP	HYBIND	HYBSUBPOP	SCA	RAO1	RAO3	RAO2	Jaya
F22	TLBO	HYBSUBPOP	HYBPOP	HYBIND	RAO3	RAO2	Jaya	SCA	CJaya	RAO1
F23	SCA	HYBIND	HYBPOP	RAO2	CJaya	Jaya	RAO3	TLBO		
F24	CJaya	HYBPOP	TLBO	HYBIND	RAO3	HYBSUBPOP	SCA	Jaya	RAO1	
F25	RAO1	Jaya	RAO2	HYBPOP	RAO3	HYBIND	HYBSUBPOP	SCA		
F26	HYBPOP	HYBIND	HYBSUBPOP	TLBO	Jaya	SCA	RAO2	CJaya	RAO1	RAO3
F27	TLBO	HYBIND								
F28	TLBO	HYBPOP	HYBIND	HYBSUBPOP	SCA					



(**a**) F1 (**b**) F8

(**c**) F11 (**d**) F18

Figure 2. Convergence curves. The population size is set as 140 and the number of iterations to 100.

Table 7. Computational times (s.) for 50 runs. The population size is set as 140 and *max_ITs* is set to 50,000.

	Jaya	CJaya	SCA	RAO1	RAO2	RAO3	TLBO	HYBSUBPOP	HYBPOP	HYBIND
F1	275.0	3263.8	1355.0	178.0	306.5	785.6	1515.0	743.2	1346.7	856.2
F2	300.5	3050.8	1406.5	193.3	324.2	811.8	764.9	777.9	1394.7	861.1
F3	50.5	231.5	105.0	43.5	55.6	55.7	44.0	92.3	89.7	85.1
F4	48.5	233.4	120.9	41.8	53.1	60.6	45.9	104.1	91.2	94.9
F5	95.3	184.8	157.2	90.4	99.7	93.8	97.9	69.5	53.8	121.5
F6	39.5	381.4	137.3	25.0	49.6	49.6	25.7	97.7	107.7	97.2
F7	58.8	544.8	221.1	33.8	68.8	68.0	33.1	144.2	157.9	138.7
F8	94.6	898.2	370.2	58.5	108.8	108.7	52.6	238.5	258.1	228.1
F9	140.4	932.0	583.1	119.3	151.5	311.0	311.0	312.9	328.0	313.7
F10	411.4	3435.2	1732.1	291.7	444.8	425.5	748.5	889.7	1446.7	803.1
F11	291.9	2652.7	1023.1	182.3	315.4	313.5	153.9	688.9	752.0	661.2
F12	48.8	457.8	186.0	30.8	61.0	59.5	28.7	122.8	130.2	118.4
F13	1760.6	1933.4	1850.7	1638.7	1619.4	1766.0	1738.0	1745.4	2057.6	1801.4
F14	33.6	210.9	94.2	26.7	37.4	36.9	28.9	71.6	72.9	69.9
F15	34.9	208.1	91.9	26.6	39.4	38.5	28.6	99.8	71.0	68.0
F16	20.2	187.3	75.8	12.7	25.6	28.3	15.1	56.2	54.9	52.4
F17	93.3	307.2	247.7	87.4	101.9	98.8	97.2	212.3	165.1	159.9
F18	295.7	763.3	615.6	281.3	296.2	296.0	263.3	683.1	442.5	472.1
F19	31.5	200.4	89.5	23.9	36.9	32.5	26.2	89.6	64.6	62.4
F20	31.8	198.4	79.0	22.1	37.7	36.4	23.4	75.2	63.8	60.2
F21	23.2	196.0	79.1	15.1	31.6	31.0	17.4	60.9	58.1	57.9
F22	305.9	650.5	443.7	298.8	319.5	312.6	295.2	388.9	426.4	382.9
F23	79.7	342.2	166.2	67.6	85.1	89.3	66.6	126.4	138.3	128.4
F24	197.3	1381.3	620.6	143.5	351.1	205.8	124.8	618.4	427.2	433.5
F25	4336.8	3192.1	1694.5	4063.5	4614.0	4018.7	1470.0	2253.0	4266.2	1493.1
F26	161.7	360.4	244.9	153.7	166.4	167.2	159.9	209.7	226.8	206.5
F27	191.5	651.4	361.7	198.1	194.4	197.9	174.5	306.0	316.4	298.1
F28	561.6	969.3	697.7	533.4	553.5	559.0	514.1	638.8	698.8	617.7

Table 8. Contribution of each algorithm to the replacements of the individuals (NoR-AI). The population size is set as 140.

	HYBPOP							HYBIND						
	Jaya	**CJaya**	**SCA**	**RAO1**	**RAO2**	**RAO3**	**TLBO**	**Jaya**	**CJaya**	**SCA**	**RAO1**	**RAO2**	**RAO3**	**TLBO**
F1	25,125	14,555	96,302	7	17,725	332,520	111,730	51,946	9347	43,566	5	25,863	4901	58,819
	(224)	(80)	(832)	(3)	(165)	(3125)	(566)	(201)	(683)	(1647)	(2)	(189)	(709)	(2781)
F2	25,246	12,585	96,155	9	17,679	333,194	98,347	51,874	9051	43,800	6	25,999	5621	57,287
	(153)	(3435)	(691)	(3)	(187)	(2382)	(24,920)	(204)	(2019)	(1249)	(1)	(196)	(916)	(8893)
F3	2873	156	182	323	2594	6666	6861	613	288	185	49	625	17,344	2543
	(51)	(9)	(24)	(53)	(53)	(208)	(120)	(46)	(25)	(14)	(10)	(50)	(8810)	(65)
F4	1368	232	511	21	1374	2348	3590	633	334	275	15	602	55,789	1321
	(31)	(12)	(29)	(4)	(40)	(239)	(87)	(43)	(20)	(18)	(4)	(41)	(19,009)	(40)
F5	4483	1028	39,631	33	4705	52,409	1444	11,980	543	19,085	29	12,800	1346	15,300
	(164)	(134)	(434)	(4)	(189)	(1294)	(200)	(71)	(219)	(141)	(5)	(54)	(247)	(170)
F6	1137	285	259	89	780	146,009	31,692	385	366	785	43	447	4170	14,094
	(58)	(14)	(34)	(14)	(43)	(7004)	(1833)	(73)	(40)	(256)	(8)	(62)	(1343)	(873)
F7	2909	253	135	873	3851	5720	5898	567	419	737	32	1195	60,536	8801
	(60)	(14)	(18)	(95)	(56)	(183)	(117)	(44)	(29)	(116)	(9)	(53)	(14,046)	(1958)
F8	3570	369	180	840	4777	13,342	15,701	672	410	1112	22	1162	20,321	69,118
	(93)	(25)	(23)	(111)	(125)	(565)	(760)	(50)	(39)	(164)	(5)	(89)	(13,307)	(2852)
F9	9658	7123	64165	116	2912	216,548	23,519	43,762	4450	28,801	76	11,319	4487	25,089
	(145)	(4361)	(451)	(11)	(67)	(3530)	(14,231)	(570)	(2101)	(120)	(11)	(91)	(871)	(3527)
F10	1055	13,634	27,550	51	198	535,232	109,601	2284	12,827	23,845	37	267	27,714	55,571
	(73)	(3278)	(741)	(8)	(13)	(7868)	(25,934)	(233)	(2881)	(1573)	(7)	(35)	(10,418)	(8180)
F11	4617	536	834	818	3734	124,739	301,758	1165	494	4006	3	530	5319	123,365
	(1330)	(22)	(360)	(442)	(1758)	(38,358)	(42,543)	(52)	(13)	(80)	(1)	(27)	(1354)	(553)
F12	3825	393	421	488	5401	6225	19,531	611	482	1090	30	863	3911	9347
	(1912)	(23)	(69)	(438)	(1662)	(3380)	(4751)	(164)	(32)	(32)	(7)	(176)	(1281)	(1420)
F13	43	409	1617	2	1530	8015	4041	256	627	663	3	973	32,210	1423
	(5)	(29)	(71)	(1)	(83)	(1388)	(512)	(16)	(39)	(25)	(1)	(35)	(18,471)	(95)
F14	1100	91	239	114	985	542	3217	468	243	481	55	413	779	1319
	(456)	(8)	(102)	(20)	(387)	(106)	(75)	(101)	(37)	(39)	(10)	(73)	(107)	(219)
F15	670	926	2581	38	651	3111	1274	784	498	1340	29	785	510	1209
	(35)	(100)	(55)	(5)	(33)	(109)	(141)	(31)	(54)	(29)	(4)	(29)	(68)	(28)
F16	272	142	155	120	4817	10,285	6404	201	400	246	45	920	45,629	2258
	(15)	(9)	(14)	(18)	(105)	(384)	(93)	(18)	(24)	(17)	(7)	(66)	(10,723)	(46)
F17	1614	80	62	504	1609	2146	3008	559	195	168	51	511	69,441	1290
	(53)	(4)	(9)	(28)	(36)	(162)	(40)	(26)	(11)	(22)	(6)	(32)	(12,934)	(42)
F18	2931	85	64	222	4076	4234	10,162	351	219	906	52	461	1122	4273
	(747)	(7)	(9)	(67)	(996)	(1275)	(2047)	(78)	(23)	(24)	(5)	(81)	(132)	(966)
F19	512	975	3020	38	489	2550	1368	818	508	1398	31	818	543	1252
	(54)	(63)	(130)	(6)	(66)	(291)	(93)	(22)	(68)	(22)	(7)	(22)	(28)	(33)
F20	594	985	2289	38	531	4788	1270	857	500	1199	29	769	816	1199
	(32)	(66)	(71)	(7)	(21)	(171)	(89)	(29)	(76)	(41)	(7)	(30)	(103)	(33)
F21	164	189	302	209	2708	8114	3782	143	531	281	37	1078	10,618	1440
	(10)	(14)	(27)	(18)	(99)	(480)	(97)	(18)	(16)	(18)	(9)	(139)	(6726)	(50)
F22	380	66	558	240	378	866	5278	247	89	415	114	242	740	4710
	(23)	(5)	(40)	(25)	(41)	(85)	(1041)	(27)	(14)	(18)	(17)	(25)	(47)	(909)
F23	1885	62	45	689	2101	2698	3214	664	214	214	48	571	97,302	1593
	(43)	(6)	(5)	(49)	(42)	(166)	(55)	(54)	(24)	(17)	(9)	(36)	(14,611)	(93)
F24	6959	1882	7941	27	5402	50,267	7073	4512	1222	5623	24	2781	532	4525
	(200)	(24)	(216)	(3)	(151)	(786)	(106)	(105)	(18)	(65)	(5)	(63)	(116)	(91)
F25	6308	368	164	2774	21,124	28,995	29,346	774	737	4481	4	1835	7938	82,428
	(1141)	(23)	(19)	(647)	(4220)	(2630)	(4062)	(33)	(37)	(74)	(3)	(102)	(1834)	(9703)
F26	1524	122	306	146	1496	1971	3772	364	333	480	77	342	17,529	1563
	(28)	(10)	(44)	(15)	(35)	(155)	(322)	(181)	(38)	(132)	(17)	(162)	(24,235)	(199)
F27	1488	77	869	311	1909	3090	7708	290	193	884	103	378	1158	3392
	(1027)	(6)	(802)	(138)	(1379)	(2155)	(1192)	(46)	(47)	(51)	(13)	(95)	(403)	(613)
F28	184	100	1674	210	875	1065	27,356	165	316	1003	104	471	626	16,520
	(18)	(9)	(225)	(37)	(130)	(192)	(4070)	(22)	(54)	(37)	(10)	(71)	(62)	(6926)

Table 9. Last iteration in which a replacement of any individual occurs (LtI-AI). The population size is set as 140.

	HYBPOP							HYBIND						
	Jaya	**CJaya**	**SCA**	**RAO1**	**RAO2**	**RAO3**	**TLBO**	**Jaya**	**CJaya**	**SCA**	**RAO1**	**RAO2**	**RAO3**	**TLBO**
F1	27,706	15,952	27,672	8	27,687	27,638	15,974	41,978	16,719	17,985	6	41,958	12,560	16,529
	(133)	(78)	(116)	(4)	(139)	(111)	(86)	(123)	(825)	(439)	(3)	(122)	(642)	(841)
F2	27,598	14,205	27,566	7	27,585	27,535	14,220	41,855	16,147	18,153	6	41,851	12,747	16,037
	(133)	(3194)	(131)	(4)	(133)	(144)	(3210)	(401)	(3033)	(471)	(3)	(401)	(940)	(2917)
F3	1115	158	192	1077	1115	1091	1092	49,975	36,436	48,752	171	49,885	49,959	1642
	(53)	(43)	(60)	(52)	(50)	(56)	(54)	(36)	(7356)	(3444)	(103)	(142)	(63)	(67)
F4	668	138	485	138	671	660	533	49,892	27,089	49,998	53	49,917	49,998	570
	(92)	(17)	(93)	(137)	(93)	(93)	(16)	(207)	(11,247)	(1)	(18)	(191)	(2)	(26)
F5	8136	301	8124	24	8146	8100	297	7280	267	6607	25	7281	4726	2744
	(83)	(42)	(76)	(13)	(75)	(70)	(41)	(154)	(97)	(109)	(10)	(155)	(627)	(52)
F6	49,889	1032	853	4730	49,915	49,740	46,411	49,059	48,335	49,999	1034	49,763	49,816	47,269
	(109)	(448)	(186)	(3277)	(101)	(493)	(1886)	(1626)	(1169)	(0)	(907)	(415)	(230)	(2201)
F7	42,814	120	80	41,573	45,096	45,055	40,202	49,995	43,638	49,999	59	49,996	49,998	46,348
	(5594)	(18)	(9)	(9922)	(3793)	(4291)	(4886)	(4)	(4013)	(0)	(23)	(2)	(1)	(3456)
F8	48,715	320	256	46,062	47,122	48,564	49,177	49,990	45,549	49,999	16	49,993	49,997	49,803
	(1000)	(47)	(52)	(3367)	(2979)	(1758)	(742)	(9)	(3461)	(0)	(4)	(4)	(2)	(290)
F9	30,943	4732	30,905	67	30,884	30,842	4733	49,998	4623	18,878	65	49,998	11,271	6159
	(525)	(2638)	(536)	(17)	(536)	(530)	(2642)	(1)	(2084)	(479)	(16)	(1)	(963)	(1442)
F10	49,434	27,375	49,999	44	46,704	49,999	27,435	49,987	29,386	49,999	45	49,946	49,780	29,077
	(873)	(6641)	(0)	(14)	(2947)	(0)	(6661)	(26)	(6347)	(0)	(10)	(62)	(206)	(5971)
F11	14,618	2561	40,446	13,232	14,525	49,776	49,999	49,683	24,945	49,999	7	49,608	46,959	49,999
	(1006)	(745)	(19,107)	(8804)	(13,913)	(563)	(0)	(135)	(10,461)	(0)	(5)	(197)	(426)	(0)
F12	9357	302	5310	5055	10,953	14,699	9658	49,072	43,353	49,999	443	49,822	49,922	8679
	(7181)	(45)	(81)	(4578)	(6044)	(14,643)	(5204)	(1526)	(6153)	(0)	(178)	(375)	(190)	(13,743)
F13	14,277	1063	14,850	1830	46,409	47,589	46,804	16,920	49,248	49,999	703	49,922	49,998	39259
	(2793)	(102)	(2653)	(5416)	(3263)	(3459)	(4071)	(2372)	(721)	(0)	(690)	(70)	(1)	(8692)
F14	30447	174	30,620	627	30428	27404	2988	49,507	43,442	49,999	359	49,513	49,870	3924
	(20,306)	(88)	(22,023)	(873)	(20,332)	(19,241)	(1819)	(780)	(5080)	(0)	(768)	(782)	(240)	(6743)
F15	303	181	302	37	303	297	183	308	167	295	28	307	285	184
	(6)	(6)	(6)	(17)	(7)	(7)	(10)	(20)	(9)	(25)	(21)	(19)	(22)	(7)
F16	267	82	110	157	1631	1527	863	25,143	48,481	49,996	64	49,897	49,970	888
	(60)	(15)	(21)	(43)	(72)	(71)	(20)	(13,492)	(1381)	(6)	(17)	(145)	(81)	(34)
F17	4402	72	64	2665	4278	3223	3444	49,989	30,269	49,998	75	49,987	49,998	16,011
	(1252)	(18)	(15)	(595)	(1104)	(729)	(707)	(13)	(7299)	(0)	(24)	(17)	(1)	(5032)
F18	44,394	6650	288	43,185	48,625	46,910	46,650	49,367	45,600	49,999	736	49,354	49,679	26,117
	(5789)	(12519)	(205)	(5242)	(986)	(1507)	(3721)	(677)	(4301)	(0)	(1541)	(658)	(319)	(11,794)
F19	385	195	375	30	380	359	194	479	174	447	37	474	435	192
	(27)	(13)	(31)	(6)	(26)	(25)	(5)	(101)	(16)	(97)	(26)	(99)	(94)	(6)
F20	663	211	640	39	653	615	209	708	210	633	52	694	623	245
	(34)	(21)	(30)	(18)	(16)	(21)	(14)	(62)	(25)	(70)	(32)	(66)	(70)	(17)
F21	425	158	529	41,474	48,520	47,210	45,157	8538	49,320	49,767	51	49,443	49,940	47,027
	(69)	(30)	(106)	(8300)	(1006)	(1836)	(4091)	(12362)	(529)	(439)	(27)	(677)	(85)	(2504)
F22	44,939	15,050	49,999	25,381	46,460	44,796	49,990	47,536	44,069	49,999	18,746	46,949	47,932	49,985
	(4800)	(11,628)	(0)	(15,057)	(5448)	(5873)	(8)	(2663)	(4699)	(0)	(13,839)	(3518)	(2883)	(15)
F23	443	59	34	355	472	421	435	49,995	39,615	49,999	58	49,991	49,998	1276
	(35)	(13)	(4)	(29)	(52)	(26)	(40)	(3)	(9172)	(0)	(29)	(8)	(1)	(274)
F24	22,208	1723	25,856	11	13,680	4089	1431	17,557	1549	24,593	3479	12,641	9469	43,818
	(14,958)	(151)	(1126)	(4)	(13,093)	(85)	(170)	(14,169)	(127)	(1456)	(10,405)	(3699)	(3695)	(7627)
F25	12,972	499	1683	8051	13,175	11,759	12,305	49,942	37,071	49,999	10,730	49,992	49,990	43,810
	(13,910)	(468)	(66)	(6134)	(14,209)	(11,152)	(13,043)	(56)	(2584)	(0)	(16,943)	(5)	(10)	(8205)
F26	4182	2569	3561	2341	4211	4082	3674	46,010	45,348	49,999	1303	47,091	49,816	13,990
	(1571)	(1527)	(1527)	(1990)	(1552)	(1569)	(1588)	(7701)	(6555)	(0)	(2433)	(3918)	(277)	(9456)
F27	26,951	344	25,400	9212	23,800	30,524	31,150	43,480	44,977	49,999	9953	47,560	46,582	27,162
	(17,884)	(92)	(18,353)	(3658)	(16,455)	(17,310)	(16,665)	(6458)	(6360)	(0)	(14,823)	(2370)	(4419)	(21,555)
F28	20,387	641	49,999	7204	45,865	47,674	42,103	23,923	48,894	49,999	2436	43,178	45,987	45,473
	(12935)	(559)	(0)	(3331)	(4309)	(1666)	(14,129)	(9779)	(900)	(0)	(4258)	(5956)	(2088)	(13,321)

Finally, Table 10 shows the last iteration in which each algorithm obtains a new optimum. A careful analysis of the results in Table 10 reveals that in the HYBPOP algorithm, the seven algorithms contribute similarly to reaching a better solution as new populations are produced. By contrast, when using the HYBIND algorithm, the powerful algorithms are CJaya and TLBO. It should be noted that the CJaya algorithm extracts random individuals from the population to generate new individuals. The TLBO algorithm collects all the individuals of the population to obtain new individuals. Therefore, these algorithms exploit the results obtained from the rest of the algorithms

to converge towards the optimum. This fact is due to the nature of these algorithms, where the best solution correctly guided the individuals. The mean value of the standard deviation is high because the LtI-BI is strongly affected by randomness behavior.

Table 10. Last iteration in which a replacement of the best individual occurs (LtI-BI). The population size is set as 140.

	HYBPOP							HYBIND						
	Jaya	**CJaya**	**SCA**	**RAO1**	**RAO2**	**RAO3**	**TLBO**	**Jaya**	**CJaya**	**SCA**	**RAO1**	**RAO2**	**RAO3**	**TLBO**
F1	0	15,863	0	0	0	0	15,730	0	15,946	14,473	0	0	0	15,509
	(0)	(84)	(0)	(0)	(0)	(0)	(116)	(0)	(1848)	(4866)	(0)	(0)	(0)	(2075)
F2	0	12,473	0	0	0	0	12,344	0	15,348	12,873	0	0	0	15,188
	(0)	(5550)	(0)	(0)	(0)	(0)	(5609)	(0)	(4498)	(6459)	(0)	(0)	(0)	(4560)
F3	475	33	2	408	329	765	943	6	16	7	2	0	31	1288
	(382)	(31)	(2)	(367)	(404)	(275)	(55)	(15)	(13)	(7)	(6)	(0)	(26)	(48)
F4	1	48	19	7	0	16	409	0	27	27	4	0	26	450
	(2)	(27)	(18)	(11)	(0)	(19)	(16)	(0)	(15)	(18)	(10)	(0)	(26)	(20)
F5	0	38	10	0	0	0	17	0	68	21	0	0	4	8
	(0)	(33)	(15)	(0)	(0)	(0)	(19)	(0)	(62)	(20)	(0)	(0)	(7)	(13)
F6	24	96	6	3	21	6850	32134	4	15	72	39	0	85	35,901
	(41)	(143)	(19)	(5)	(42)	(2653)	(1169)	(9)	(14)	(93)	(89)	(0)	(112)	(1651)
F7	1764	23	11	1941	1077	5315	5811	0	21	2	10	2	46	24,451
	(1964)	(24)	(22)	(2372)	(1547)	(7355)	(8283)	(0)	(18)	(4)	(23)	(6)	(34)	(11,193)
F8	5681	78	0	8600	8735	13,766	18,316	0	45	0	0	0	193	44,652
	(8188)	(33)	(0)	(9945)	(12,438)	(13,950)	(11,867)	(0)	(16)	(0)	(0)	(0)	(329)	(5431)
F9	0	3322	0	0	0	1	3241	0	3308	0	3	0	1	3086
	(0)	(2657)	(0)	(0)	(0)	(2)	(2667)	(0)	(1939)	(0)	(5)	(0)	(0)	(1971)
F10	0	25,452	0	0	0	1	25,355	0	27,404	7926	0	0	0	27,330
	(0)	(9341)	(0)	(0)	(0)	(2)	(9451)	(0)	(7900)	(12,114)	(0)	(0)	(0)	(7991)
F11	10,586	1241	0	12,384	8518	29,734	49,977	0	357	0	0	0	0	49,996
	(11,106)	(809)	(0)	(14,485)	(8532)	(6041)	(28)	(0)	(109)	(0)	(0)	(0)	(0)	(4)
F12	750	84	22	867	741	758	2766	17	125	16	84	20	250	4035
	(753)	(40)	(44)	(760)	(793)	(752)	(812)	(49)	(96)	(32)	(128)	(60)	(101)	(679)
F13	1	330	71	1	1	3	8260	1	476	374	2	1	2	14,327
	(1)	(158)	(86)	(1)	(3)	(4)	(5143)	(1)	(228)	(162)	(5)	(1)	(3)	(8720)
F14	11	5	0	1	1	5	414	0	5	2	1	1	6	560
	(22)	(7)	(0)	(2)	(2)	(9)	(45)	(0)	(11)	(3)	(1)	(1)	(7)	(126)
F15	0	62	10	0	0	0	25	0	41	22	0	0	2	6
	(0)	(50)	(17)	(0)	(0)	(0)	(31)	(0)	(28)	(35)	(0)	(0)	(3)	(8)
F16	1	29	4	23	9	19	705	1	14	2	8	1	23	753
	(2)	(23)	(8)	(39)	(12)	(20)	(6)	(1)	(10)	(3)	(24)	(1)	(16)	(35)
F17	168	5	0	154	114	195	239	2	1	0	0	1	6	645
	(96)	(13)	(0)	(70)	(97)	(57)	(15)	(4)	(1)	(0)	(0)	(1)	(8)	(188)
F18	6493	6472	5	6667	8367	7576	8274	3	2	0	534	8	3	14,666
	(12,607)	(12,609)	(14)	(12,188)	(11,997)	(12,953)	(12,092)	(6)	(4)	(0)	(1594)	(18)	(4)	(12,688)
F19	1	59	8	0	0	2	11	0	66	51	0	0	6	12
	(1)	(42)	(9)	(0)	(0)	(3)	(21)	(0)	(42)	(45)	(0)	(0)	(10)	(23)
F20	0	49	1	0	0	1	37	0	61	34	0	0	0	37
	(0)	(43)	(1)	(0)	(0)	(3)	(41)	(0)	(47)	(22)	(0)	(0)	(0)	(58)
F21	1	47	8	8820	6337	9098	5499	1	32	10	2	0	4850	16,989
	(3)	(24)	(13)	(5982)	(15,046)	(16,086)	(12,564)	(1)	(26)	(15)	(4)	(0)	(14,515)	(17,876)
F22	88	276	48	55	19	197	44183	3	0	0	74	39	99	46,093
	(164)	(827)	(104)	(84)	(42)	(166)	(12,373)	(8)	(0)	(0)	(183)	(81)	(139)	(7310)
F23	211	5	0	177	151	213	247	1	8	0	6	1	3	564
	(36)	(7)	(0)	(64)	(87)	(46)	(11)	(2)	(10)	(0)	(12)	(1)	(4)	(46)
F24	0	1034	0	0	0	0	786	0	1144	0	0	0	0	757
	(0)	(74)	(0)	(0)	(0)	(0)	(203)	(0)	(85)	(0)	(0)	(0)	(0)	(177)
F25	3665	232	0	3948	3720	3687	3792	0	116	4662	10,704	0	0	43,752
	(679)	(134)	(0)	(312)	(479)	(456)	(485)	(0)	(45)	(13,984)	(16,898)	(0)	(0)	(8269)
F26	26	59	2	6	9	5	632	11	59	9	11	10	20	900
	(32)	(52)	(4)	(14)	(11)	(8)	(141)	(25)	(67)	(20)	(22)	(19)	(36)	(422)
F27	501	16	5004	328	350	2802	13,485	4058	78	20,001	27	253	2746	11,151
	(919)	(32)	(14,995)	(483)	(477)	(5567)	(17,667)	(12,128)	(87)	(24,493)	(71)	(752)	(8058)	(16,657)
F28	2	36	2	18	4	9	11474	0	13	3	30	11	16	11,993
	(2)	(67)	(3)	(22)	(7)	(11)	(8711)	(0)	(23)	(5)	(39)	(24)	(15)	(7744)

It has been found that the HYBSUBPOP algorithm does not reach excellent optimization performance because of the lack of harmony between the original algorithms, so it has left without further analysis. On the other hand, the exploitation phase of the HYBPOP and HYBIND algorithms are similar. In contrast, the HYBIND algorithm outperforms the HYBPOP one in terms of exploitation.

The hybridization of the original algorithms is implemented at the individual level in the HYBIND algorithm, contrary to the HYBPOP algorithm, in which that hybridization is performed at the population level. Finally, the HYBPOP algorithm included algorithms that update the population without analyzing the fitness of the associated solutions, while this restriction is mandatory in the HYBIND algorithm.

5. Conclusions

This paper proposed a hybridization strategy of seven well-known algorithms. Three hybrid algorithms free of setting parameters dubbed the HYBSUBPOP, HYBPOP, and HYBIND algorithms are designed. These algorithms are derived from a dynamic skeleton allowing the inclusion of any metaheuristic optimization algorithm that exhibits further improvements. The only requirement in merging a new optimization algorithm into the proposed skeleton is to know if the replacement of an individual on that algorithm is based on the enhancement of the cost function or not. Moreover, both chaotic algorithms and multi-phase algorithms have been employed to design the proposed hybrid algorithms, which proves the versatility of the proposed hybridization skeleton. The experimental results show that the HYBPOP and HYBIND algorithms effectively exploit the capabilities of all the considered algorithms. They present an excellent ability to solve a large number of benchmark functions while improving the quality of the solutions obtained. Generally speaking, the hybridization at the individual level is better than that at the population level, which explains why the performance of the HYBSUBPOP algorithm is inferior to the other hybrid algorithms. As future lines of work, we intend to integrate more efficient algorithms into the proposed hybridization skeleton as well as to evaluate new versions of hybridization, and extend the performance analysis of the potential algorithms for solving more complex functions and real-world engineering problems.

Author Contributions: H.M., A.B., J.-L.S.-R., H.R., and A.J.-M. conceived the hybrid algorithms; A.B. conceived the Chaotic 2D Jaya algorithm; H.M. designed the hybrid algorithms; H.M. codified the hybrid algorithms; H.M., J.-L.S.-R., and H.R. performed numerical experiments; H.M., A.B., and A.J.-M. analyzed the data; H.M. wrote the original draft. A.B., J.-L.S.-R., H.R., and A.J.-M. reviewed and edited the manuscript. All the authors have read and agreed to the published version of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research and APC was funded by the Spanish Ministry of Science, Innovation and Universities and the Research State Agency under Grant RTI2018-098156-B-C54 co-financed by FEDER funds, and by the Spanish Ministry of Economy and Competitiveness under Grant TIN2017-89266-R, co-financed by FEDER funds.

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

References

1. Sadollah, A.; Bahreininejad, A.; Eskandar, H.; Hamdi, M. Mine Blast Algorithm: A New Population Based Algorithm for Solving Constrained Engineering Optimization Problems. *Appl. Soft Comput.* **2013**, *13*, 2592–2612 [CrossRef]
2. Zhao, W.; Zhang, Z.; Wang, L. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Eng. Appl. Artif. Intell.* **2020**, *87*, 103300. [CrossRef]
3. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Comput. Struct.* **2016**, *169*, 1–12. [CrossRef]
4. Dorigo, M.; Di Caro, G. *New Ideas in Optimization*; Chapter The Ant Colony Optimization Meta-Heuristic; McGraw-Hill Ltd.: Maidenhead, UK, 1999; pp. 11–32.
5. Ma, H.; Simon, D.; Siarry, P.; Yang, Z.; Fei, M. Biogeography-Based Optimization: A 10-Year Review. *IEEE Trans. Emerg. Top. Comput. Intell.* **2017**, *1*, 391–407. [CrossRef]
6. Ahrari, A.; Atai, A.A. Grenade Explosion Method—A novel tool for optimization of multimodal functions. *Appl. Soft Comput.* **2010**, *10*, 1132–1140. [CrossRef]
7. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intell.* **2007**, *1*, 33–57. [CrossRef]
8. Xin-She, Y. Firefly Algorithm, Lévy Flights and Global Optimization. *Res. Dev. Intell. Syst. XXVI* **2009**, 209–218. [CrossRef]

9. Karaboga, D.; Basturk, B. On the Performance of Artificial Bee Colony (ABC) Algorithm. *Appl. Soft Comput.* **2008**, *8*, 687–697. [CrossRef]
10. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
11. Eusuff, M.; Lansey, K.; Pasha, F. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Eng. Optim.* **2006**, *38*, 129–154. [CrossRef]
12. Szénási, S.; Felde, I. Configuring genetic algorithm to solve the inverse heat conduction problem. *Acta Polytech. Hung.* **2017**, *14*, 133–152. [CrossRef]
13. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
14. Rao, R.V.; Savsani, V.; Vakharia, D. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput.-Aided Des.* **2011**, *43*, 303–315. [CrossRef]
15. Zhao, W.; Wang, L.; Zhang, Z. Supply-Demand-Based Optimization: A Novel Economics-Inspired Algorithm for Global Optimization. *IEEE Access* **2019**, *7*, 73182–73206. [CrossRef]
16. Rao, R.V. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 19–34. [CrossRef]
17. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
18. Rao, R.V. Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems. *Int. J. Ind. Eng. Comput.* **2020**, *11*, 107–130. [CrossRef]
19. Majumdar, M.; Mitra, T.; Nishimura, K. *Optim. Chaos*; Springer: New York, NY, USA, 2000.
20. Ott, E. Frontmatter. In *Chaos in Dynamical Systems*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2002; pp. i–iv.
21. Gandomi, A.; Yang, X.S.; Talatahari, S.; Alavi, A. Firefly algorithm with chaos. *Commun. Nonlinear Sci. Numer. Simul.* **2013**, *18*, 89–98. [CrossRef]
22. Gokhale, S.; Kale, V. An application of a tent map initiated Chaotic Firefly algorithm for optimal overcurrent relay coordination. *Int. J. Electr. Power Energy Syst.* **2016**, *78*, 336–342. [CrossRef]
23. Ma, Z.S. Chaotic populations in genetic algorithms. *Appl. Soft Comput.* **2012**, *12*, 2409–2424. [CrossRef]
24. Yan, X.F.; Chen, D.Z.; Hu, S.X. Chaos-genetic algorithms for optimizing the operating conditions based on RBF-PLS model. *Comput. Chem. Eng.* **2003**, *27*, 1393–1404. [CrossRef]
25. Hong, W.C. Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm. *Neurocomputing* **2011**, *74*, 2096–2107. [CrossRef]
26. Mingjun, J.; Huanwen, T. Application of chaos in simulated annealing. *Chaos Solitons Fractals* **2004**, *21*, 933–941. [CrossRef]
27. Saremi, J.; Mirjalili, S.; Lewisn, A. Biogeography-based optimisation with chaos. *Neural Comput. Appl.* **2014**, *25*, 1077–1097. [CrossRef]
28. Wang, X.; Duan, H. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Comput. Ind. Eng.* **2014**, *73*, 96–114. [CrossRef]
29. Jia, D.; Zheng, G.; Khan, M.K. An effective memetic differential evolution algorithm based on chaotic local search. *Inf. Sci.* **2011**, *181*, 3175–3187. [CrossRef]
30. Peng, C.; Sun, H.; Guo, J.; Liu, G. Dynamic economic dispatch for wind-thermal power system using a novel bi-population chaotic differential evolution algorithm. *Int. J. Electr. Power Energy Syst.* **2012**, *42*, 119–126. [CrossRef]
31. Alatas, B. Chaotic bee colony algorithms for global numerical optimization. *Expert Syst. Appl.* **2010**, *37*, 5682–5687. [CrossRef]
32. Yu, J.; Kim, C.H.; Wadood, A.; Khurshiad, T.; Rhee, S.B. A Novel Multi-Population Based Chaotic JAYA Algorithm with Application in Solving Economic Load Dispatch Problems. *Energies* **2018**, *11*, 1946. [CrossRef]
33. Farah, A.; Belazi, A. A novel chaotic Jaya algorithm for unconstrained numerical optimization. *Nonlinear Dyn.* **2018**, *93*, 1451–1480. [CrossRef]
34. Kumar, Y.; Singh, P.K. RA chaotic teaching learning based optimization algorithm for clustering problems. *Appl. Intell.* **2019**, *49*, 107–130. [CrossRef]
35. Moayedi, H.; Gör, M.; Khari, M.; Foong, L.K.; Bahiraei, M.; Bui, D.T. Hybridizing four wise neural-metaheuristic paradigms in predicting soil shear strength. *Measurement* **2020**, *156*, 107576. [CrossRef]

36. Nguyen, B.M.; Tran, T.; Nguyen, T.; Nguyen, G. Hybridization of Galactic Swarm and Evolution Whale Optimization for Global Search Problem. *IEEE Access* **2020**, *8*, 74991–75010. [CrossRef]
37. Kayabekir, A.E.; Toklu, Y.C.; Bekdaş, G.; Nigdeli, S.M.; Yücel, M.; Geem, Z.W. A Novel Hybrid Harmony Search Approach for the Analysis of Plane Stress Systems via Total Potential Optimization. *Appl. Sci.* **2020**, *10*, 2301. [CrossRef]
38. Punurai, W.; Azad, M.S.; Pholdee, N.; Bureerat, S.; Sinsabvarodom, C. A novel hybridized metaheuristic technique in enhancing the diagnosis of cross-sectional dent damaged offshore platform members. *Comput. Intell.* **2020**, *36*, 132–150. [CrossRef]
39. Pellegrini, R.; Serani, A.; Liuzzi, G.; Rinaldi, F.; Lucidi, S.; Diez, M. Hybridization of Multi-Objective Deterministic Particle Swarm with Derivative-Free Local Searches. *Mathematics* **2020**, *8*, 546. [CrossRef]
40. Yue, Z.; Zhang, S.; Xiao, W. A Novel Hybrid Algorithm Based on Grey Wolf Optimizer and Fireworks Algorithm. *Sensors* **2020**, *20*, 2147. [CrossRef]
41. Seifi, A.; Ehteram, M.; Singh, V.P.; Mosavi, A. Modeling and Uncertainty Analysis of Groundwater Level Using Six Evolutionary Optimization Algorithms Hybridized with ANFIS, SVM, and ANN. *Sustainability* **2020**, *12*, 4023. [CrossRef]
42. Chen, X.; Yu, K. Hybridizing cuckoo search algorithm with biogeography-based optimization for estimating photovoltaic model parameters. *Sol. Energy* **2019**, *180*, 192–206. [CrossRef]
43. Zhang, X.; Shen, X.; Yu, Z. A Novel Hybrid Ant Colony Optimization for a Multicast Routing Problem. *Algorithms* **2019**, *12*, 18. [CrossRef]
44. Aljohani, T.M.; Ebrahim, A.F.; Mohammed, O. Single and Multiobjective Optimal Reactive Power Dispatch Based on Hybrid Artificial Physics—Particle Swarm Optimization. *Energies* **2019**, *12*, 2333. [CrossRef]
45. Ahmadian, A.; Elkamel, A.; Mazouz, A. An Improved Hybrid Particle Swarm Optimization and Tabu Search Algorithm for Expansion Planning of Large Dimension Electric Distribution Network. *Energies* **2019**, *12*, 3052. [CrossRef]
46. Li, G.; Liu, P.; Le, C.; Zhou, B. A Novel Hybrid Meta-Heuristic Algorithm Based on the Cross-Entropy Method and Firefly Algorithm for Global Optimization. *Entropy* **2019**, *21*, 494. [CrossRef]
47. Cherki, I.; Chaker, A.; Djidar, Z.; Khalfallah, N.; Benzergua, F. A Sequential Hybridization of Genetic Algorithm and Particle Swarm Optimization for the Optimal Reactive Power Flow. *Sustainability* **2019**, *11*, 3862. [CrossRef]
48. hanem, W.A.H.M.; Jantan, A. Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems. *Neural Comput. Appl.* **2018**, *30*, 163–181. [CrossRef]
49. Das, P.; Behera, H.; Panigrahi, B. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm Evol. Comput.* **2016**, *28*, 14–28. [CrossRef]
50. Wang, G.G.; Gandomi, A.H.; Zhao, X.; Chu, H.C.E. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. [CrossRef]
51. Zhu, A.; Xu, C.; Li, Z.; Wu, J.; Liu, Z. Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC. *J. Syst. Eng. Electron.* **2015**, *26*, 317–328. [CrossRef]
52. Javaid, N.; Ahmed, A.; Iqbal, S.; Ashraf, M. Day Ahead Real Time Pricing and Critical Peak Pricing Based Power Scheduling for Smart Homes with Different Duty Cycles. *Energies* **2018**, *11*, 1464. [CrossRef]
53. Mishra, S.; Ray, P.K. Power quality improvement using photovoltaic fed DSTATCOM based on JAYA optimization. *IEEE Trans. Sustain. Energy* **2016**, *7*, 1672–1680. [CrossRef]
54. Huang, C.; Wang, L.; Yeung, R.S.; Zhang, Z.; Chung, H.S.; Bensoussan, A. A Prediction Model-Guided Jaya Algorithm for the PV System Maximum Power Point Tracking. *IEEE Trans. Sustain. Energy* **2018**, *9*, 45–55. [CrossRef]
55. Abhishek, K.; Kumar, V.R.; Datta, S.; Mahapatra, S.S. Application of JAYA algorithm for the optimization of machining performance characteristics during the turning of CFRP (epoxy) composites: comparison with TLBO, GA, and ICA. *Eng. Comput.* **2016**, 1–19. [CrossRef]
56. Choudhary, A.; Kumar, M.; Unune, D.R. Investigating effects of resistance wire heating on AISI 1023 weldment characteristics during ASAW. *Mater. Manuf. Process.* **2018**, *33*, 759–769. [CrossRef]

57. Dinh-Cong, D.; Dang-Trung, H.; Nguyen-Thoi, T. An efficient approach for optimal sensor placement and damage identification in laminated composite structures. *Adv. Eng. Softw.* **2018**, *119*, 48–59. [CrossRef]
58. Singh, S.P.; Prakash, T.; Singh, V.; Babu, M.G. Analytic hierarchy process based automatic generation control of multi-area interconnected power system using Jaya algorithm. *Eng. Appl. Artif. Intell.* **2017**, *60*, 35–44. [CrossRef]
59. Cruz, N.C.; Redondo, J.L.; Álvarez, J.D.; Berenguel, M.; Ortigosa, P.M. A parallel Teaching–Learning-Based Optimization procedure for automatic heliostat aiming. *J. Supercomput.* **2017**, *73*, 591–606. [CrossRef]
60. Rao, R.V.; Pawar, R.B. Self-adaptive Multi-population Rao Algorithms for Engineering Design Optimization. *Appl. Artif. Intell.* **2020**, *34*, 187–250. [CrossRef]
61. Rao, R.; Pawar, R. Constrained design optimization of selected mechanical system components using Rao algorithms. *Appl. Soft Comput.* **2020**, *89*, 106141. [CrossRef]
62. Rao, R.V.; Keesari, H.S. Rao algorithms for multi-objective optimization of selected thermodynamic cyclesn. *Eng. Comput.* **2020**. [CrossRef]
63. Kumar-Majhi, S. An Efficient Feed Foreword Network Model with Sine Cosine Algorithm for Breast Cancer Classification. *Int. J. Syst. Dyn. Appl. (IJSDA)* **2018**, *7*, 1–14. [CrossRef]
64. Rajesh, K.S.; Dash, S.S. Load frequency control of autonomous power system using adaptive fuzzy based PID controller optimized on improved sine cosine algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 2361–2373. [CrossRef]
65. Khezri, R.; Oshnoei, A.; Tarafdar Hagh, M.; Muyeen, S. Coordination of Heat Pumps, Electric Vehicles and AGC for Efficient LFC in a Smart Hybrid Power System via SCA-Based Optimized FOPID Controllers. *Energies* **2018**, *11*, 420. [CrossRef]
66. Ramanaiah, M.L.; Reddy, M.D. Sine Cosine Algorithm for Loss Reduction in Distribution System with Unified Power Quality Conditioner. *i-Manag. J. Power Syst. Eng.* **2017**, *5*, 10–16. [CrossRef]
67. Dhundhara, S.; Verma, Y.P. Capacitive energy storage with optimized controller for frequency regulation in realistic multisource deregulated power system. *Energy* **2018**, *147*, 1108–1128. [CrossRef]
68. Singh, V.P. Sine cosine algorithm based reduction of higher order continuous systems. In Proceedings of the 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 7–8 December 2017; pp. 649–653. [CrossRef]
69. Das, S.; Bhattacharya, A.; Chakraborty, A.K. Solution of short-term hydrothermal scheduling using sine cosine algorithm. *Soft Comput.* **2018**, *22*, 6409–6427. [CrossRef]
70. Singh, M.; Panigrahi, B.; Abhyankar, A. Optimal coordination of directional over-current relays using Teaching Learning-Based Optimization (TLBO) algorithm. *Int. J. Electr. Power Energy Syst.* **2013**, *50*, 33–41. [CrossRef]
71. Niknam, T.; Azizipanah-Abarghooee, R.; Narimani, M.R. A new multi objective optimization approach based on TLBO for location of automatic voltage regulators in distribution systems. *Eng. Appl. Artif. Intell.* **2012**, *25*, 1577–1588. [CrossRef]
72. Li, D.; Zhang, C.; Shao, X.; Lin, W. A multi-objective TLBO algorithm for balancing two-sided assembly line with multiple constraints. *J. Intell. Manuf.* **2016**, *27*, 725–739. [CrossRef]
73. Arya, L.; Koshti, A. Anticipatory load shedding for line overload alleviation using Teaching learning based optimization (TLBO). *Int. J. Electr. Power Energy Syst.* **2014**, *63*, 862–877. [CrossRef]
74. Mohanty, B. TLBO optimized sliding mode controller for multi-area multi-source nonlinear interconnected AGC system. *Int. J. Electr. Power Energy Syst.* **2015**, *73*, 872–881. [CrossRef]
75. Yan, J.; Li, K.; Bai, E.; Yang, Z.; Foley, A. Time series wind power forecasting based on variant Gaussian Process and TLBO. *Neurocomputing* **2016**, *189*, 135–144. [CrossRef]
76. Baghban, A.; Kardani, M.N.; Mohammadi, A.H. Improved estimation of Cetane number of fatty acid methyl esters (FAMEs) based biodiesels using TLBO-NN and PSO-NN models. *Fuel* **2018**, *232*, 620–631. [CrossRef]
77. Migallón, H.; Jimeno-Morenilla, A.; Sánchez-Romero, J.; Belazi, A. Efficient parallel and fast convergence chaotic Jaya algorithms. *Swarm Evol. Comput.* **2020**, 100698. [CrossRef]
78. Ravipudi, J.L.; Neebha, M. Synthesis of linear antenna arrays using Jaya, self-adaptive Jaya and chaotic Jaya algorithms. *AEU-Int. J. Electron. Commun.* **2018**, *92*, 54–63. [CrossRef]

79. Free Software Foundation, Inc. GCC, the GNU Compiler Collection. Available online: https://www.gnu.org/software/gcc/index.html (accessed on 10 March 2017).
80. García-Monzó, A.; Migallón, H.; Jimeno-Morenilla, A.; Sánchez-Romero, J.L.; Rico, H.; Rao, R.V. Efficient Subpopulation Based Parallel TLBO Optimization Algorithms. *Electronics* **2018**, *8*, 19. [CrossRef]

© 2020 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 (http://creativecommons.org/licenses/by/4.0/).