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; hmigallon@umh.es
[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; hector.rico@gmail.com; jimeno@dtic.ua.es
* Correspondence: hmigallon@umh.es; Tel.: +34-966658390
† These authors contributed equally to this work.

Version June 2, 2020 submitted to Mathematics

Abstract: Several population-based metaheuristic optimization algorithms have been proposed in the last decades, none of which is 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. These algorithms are Jaya, SCA, Rao's algorithms, 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. Besides, 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. 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) [12] based on the sine and cosine trigonometric functions; the teaching-learning-based optimization (TLBO) algorithm [13] based on the processes of teaching and learning; the supply-demand-based optimization method (SDO) [14] based on both the demand relation of consumers and supply relation of producers; the Jaya algorithm [15] based on geometric distances and random processes; the Harris haws optimization method (HHO) [16] based on the cooperative behavior and chasing style of Harris' hawks, and Rao optimization algorithms [17]; 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 [18,19]. They can be applied to replace the PRNGs in producing the control parameters or performing local searches [20–33]. However, improving an optimization algorithm using chaotic systems instead of 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 [34–51]. 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 [15], the SCA algorithm [12], the supply-demand-based optimization method [14], Rao's optimization algorithms [17], the Harris hawks optimization method (HHO) [16] and the teaching-learning-based optimization (TLBO) algorithm [13]. 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. Besides, the SDO algorithm, which offers impressive initial results, works with two populations preventing its integration in our hybrid proposals.

The Jaya optimization algorithm, described in Algorithm 1, has been successfully used for solving a large number of large-scale industrial problems [52–58].

Algorithm 1 Jaya algorithm

```
Set max_ITs
Set the 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
    Search for the current WorstPop
    for m = 0 to popSize do
        for k = 1 to numDesignVars do
            r_1 = rand_{0..1}
            r_2 = rand_{0..1}
            newPop_m^k = Pop_m^k + r_1 (BestPop^k − |Pop_m^k|) − r_2 (WorstPop^k − |Pop_m^k|)
            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
```

The SCA algorithm, presented in Algorithm 2 has been proven to be efficient in several applications [59–65].

Algorithm 2 SCA optimization algorithm

1: Set $iniValue\_r_1 = 2$
2: Set max_ITs
3: Set the population size (Iterator individuals: m)
4: Define the function cost (Iterator design variables: k)
5: Generate the initial population Pop {lines 5-11 of Algorithm 1}
6: **for** $iterator = 1$ to max_ITs **do**
7: Search for the current $BestPop$
8: $r_1 = iniValue\_r_1 - iterator(iniValue\_r_1/max\_ITs)$
9: **for** $m = 0$ to $popSize$ **do**
10: **for** $k = 1$ to $numDesignVars$ **do**
11: $r_2 = 2\pi rand_{0..1}$
12: $r_3 = 2rand_{0..1}$
13: $r_4 = rand_{0..1}$
14: **if** $r_4 < 0.5$ **then**
15: $newPop_m^k = Pop_m^k(r_1 \sin(r_2) \left| r_3 BestPop^k - Pop_m^k \right|)$
16: **else**
17: $newPop_m^k = Pop_m^k(r_1 \cos(r_2) \left| r_3 BestPop^k - Pop_m^k \right|)$
18: **end if**
19: Check the bounds of $newPop_m^k$ {lines 20-25 of Algorithm 1}
20: **end for**
21: **if** $F(newPop_m) < F(Pop_m)$ **then**
22: $Pop_m = newPop_m$ {replace the current population}
23: **end if**
24: **end for**
25: **end for**
26: Search for the current $BestPop$

The three new Rao's optimization algorithms (i.e., RAO1, RAO2, and RAO3), shown in Algorithm 3, are metaphor-less algorithms based on the best and worst solutions obtained during the optimization process and the random interactions between the candidate solutions [66–68].

Algorithm 3 Rao algorithms

1: Set max_ITs
2: Set the population size (Iterator individuals: m)
3: Define the function cost (Iterator design variables: k)
4: Generate the initial population Pop {lines 5-11 of Algorithm 1}
5: **for** $iterator = 1$ to max_ITs **do**
6: Search for the current $BestPop$
7: Search for the current $WorstPop$
8: **for** $m = 0$ to $popSize$ **do**
9: Select the random individual $RandPop \neq Pop_m$
10: **for** $k = 1$ to $numDesignVars$ **do**
11: **if** RAO1 **then**
12: $r_1 = rand_{0..1}$
13: $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - WorstPop^k \right)$
14: **end if**
15: **if** RAO2 **then**
16: $r_1 = rand_{0..1}$
17: $r_2 = rand_{0..1}$
18: **if** $Pop_m^k < RandPop^k$ **then**
19: $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - WorstPop^k \right) - r_2 \left(\left| Pop_m^k \right| - \left| RandPop^k \right| \right)$
20: **else**
21: $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - WorstPop^k \right) - r_2 \left(\left| RandPop^k \right| - \left| Pop_m^k \right| \right)$
22: **end if**
23: **end if**
24: **if** RAO3 **then**
25: $r_1 = rand_{0..1}$
26: $r_2 = rand_{0..1}$
27: **if** $Pop_m^k < RandPop^k$ **then**
28: $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - \left| WorstPop^k \right| \right) - r_2 \left(\left| Pop_m^k \right| - \left| RandPop^k \right| \right)$
29: **else**
30: $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - \left| WorstPop^k \right| \right) - r_2 \left(\left| RandPop^k \right| - \left| Pop_m^k \right| \right)$
31: **end if**
32: **end if**
33: Check the bounds of $newPop_m^k$ {lines 20-25 of Algorithm 1}
34: **end for**
35: **if** $F(newPop_m) < F(Pop_m)$ **then**
36: $Pop_m = newPop_m$ {replace the current population}
37: **end if**
38: **end for**
39: **end for**
40: Search for the current $BestPop$

The TLBO algorithm, described in Algorithm 4, is a two-phase algorithm; teacher phase and learner phase. It has been proven effective in solving various engineering problems [69–75].

Algorithm 4 TLBO algorithm

1: Set $iniValue\_r_1 = 2$
2: Set max_ITs
3: Set the population size (Iterator individuals: m)
4: Define the function cost (Iterator design variables: k)
5: Generate the initial population Pop {lines 5-11 of Algorithm 1}
6: Set $Phase = TeacherPhase$
7: **for** $iterator = 1$ to max_ITs **do**
8: Search for the current $BestPop$
9: Set the teaching factor T_F (an integer random value $\in [1, 2]$)
10: **for** $k = 1$ to $numDesignVars$ **do**
11: $AveragePop^k = \left(\sum_1^m BestPop^k\right) / numDesignVars$
12: **end for**
13: **for** $m = 0$ to $popSize$ **do**
14: Select the random individual $RandPop \neq Pop_m$
15: **for** $k = 1$ to $numDesignVars$ **do**
16: **if** $Phase = TeacherPhase$ **then**
17: $r_1 = rand_{0..1}$
18: $newPop_m^k = Pop_m^k + r_1 \left(BestPop^k - T_F AveragePop^k\right)$
19: **end if**
20: **if** $Phase = LearnerPhase$ **then**
21: $r_1 = rand_{0..1}$
22: **if** $Pop_m^k < RandPop^k$ **then**
23: $newPop_m^k = Pop_m^k + r_1 \left(Pop_m^k - RandPop^k\right)$
24: **else**
25: $newPop_m^k = Pop_m^k + r_1 \left(RandPop^k - Pop_m^k\right)$
26: **end if**
27: **end if**
28: Check the bounds of $newPop_m^k$ {lines 20-25 of Algorithm 1}
29: **end for**
30: **if** $F(newPop_m) < F(Pop_m)$ **then**
31: $Pop_m = newPop_m$ {replace the current population}
32: **end if**
33: **end for**
34: **if** $Phase = TeacherPhase$ **then**
35: $Phase = LearnerPhase$
36: **else**
37: $Phase = TeacherPhase$
38: **end if**
39: **end for**
40: Search for the current $BestPop$

As mentioned earlier, the use of chaotic maps can improve the behavior of some metaheuristic methods. The 2D chaotic map reported in [32] has significantly improved the convergence rate of the Jaya algorithm [32,76]. The generation of the 2D chaotic map is shown in Algorithm 5, 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 6, where $ch_x, x \in [1 \ldots 6]$ are chaotic values randomly extracted from the 2D chaotic map. Other chaotic maps have been applied to Jaya in [77,78]. However, they do not surmount the chaotic behavior of the aforementioned 2D map.

Algorithm 5 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 6 Chaotic 2D Jaya algorithm

Set $iniValue\_r_1 = 2$
Set max_ITs
Set the population size (Iterator individuals: m)
Define the function cost (Iterator design variables: k)
Generate the initial population Pop {lines 5-11 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 $RandPop \neq Pop_m$
    $r_1 = rand_{0..1}$
    $r_2 = rand_{0..1}$
    $r_a = min(r_1, r_2)$
    $r_b = max(r_1, r_2)$
    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 20-25 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 $BestPop$

As they present a similar structure, Algorithms 1-6 are used for designing our hybrid algorithms.

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 7 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 27-32 of Algorithm 7). The algorithm used to obtain a new individual is determined by $AlgSelected$ (see line 20 of Algorithm 7).

Algorithm 7 Skeleton of hybrid algorithms

1: Set max_ITs
2: Set the population size (Iterator individuals: m)
3: Define the function cost (Iterator design variables: k)
4: Set $iniValue\_r_1 = 2$
5: Set $Phase = TeacherPhase$; $RepeatTLBO = false$
6: Generate the initial population Pop {lines 5-11 of Algorithm 1}
7: **for** $iterator = 1$ to max_ITs **do**
8: Search for the current $BestPop$
9: Search for the current $WorstPop$
10: Set the scaling factor S_F and teaching factor T_F (an integer random value $\in [1, 2]$)
11: **for** $k = 1$ to $numDesignVars$ **do**
12: $AveragePop^k = \left(\sum_1^m BestPop^k\right) / numDesignVars$
13: **end for**
14: $r_1 = iniValue\_r_1 - iterator(iniValue\_r_1 / max\_ITs)$
15: **for** $m = 0$ to $popSize$ **do**
16: Select random individual $RandPop \neq Pop_m$
17: $r_2 = rand_{0..1}$; $r_3 = rand_{0..1}$
18: $r_a = min(r_2, r_3)$; $r_b = max(r_2, r_3)$
19: **for** $k = 1$ to $numDesignVars$ **do**
20: $\Rightarrow$ ($AlgSelected$) Compute $newPop_m^k$ using $AlgSelected$ (one from Algorithms 1-6)
21: Check the bounds of $newPop_m^k$ {lines 20-25 of Algorithm 1}
22: **end for**
23: **if** $F(newPop_m) < F(Pop_m)$ **then**
24: $Pop_m = newPop_m$ {Replace the current population}
25: **end if**
26: **if** $TLBO$ **then**
27: **if** $Phase = TeacherPhase$ **then**
28: $Phase = LearnerPhase$; $RepeatTLBO = true$;$m = m - 1$
29: **else**
30: $Phase = TeacherPhase$; $RepeatTLBO = false$
31: **end if**
32: **end if**
33: **end for**
34: **end for**
35: Search for the current $BestPop$

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 8, 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 7 is not mandatory on all algorithms.

Algorithm 8 HYBPOP: Hybrid algorithm based on population

```
NumOfAlgorithms = 7
Selection = 0
for iterator = 1 to max_ITs do
    switch (Selection)
    case 0:
    AlgSelected = Jaya
    case 1:
    AlgSelected = Chaotic Jaya
    case 2:
    AlgSelected = SCA
    case 3:
    AlgSelected = RAO1
    case 4:
    AlgSelected = RAO2
    case 5:
    AlgSelected = RAO3
    case 6:
    AlgSelected = TLBO
    end switch
    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 ≥ NumOfAlgorithms then
        Selection = 0
    end if
end for
Search for the current BestPop
```

The second algorithm, named HYBSUBPOP, is described through Algorithm 9. 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 9 HYBSUBPOP: Hybrid algorithm based on sub-populations

```
Split popSize into NumOfAlgorithms sub-populations
for iterator = 1 to max_ITs do
    for m = 0 to popSize do
        subPopID = sub-population index of individual m.
        switch (subPopID)
        case 0:
        AlgSelected = Jaya
        case 1:
        AlgSelected = Chaotic Jaya
        case 2:
        AlgSelected = SCA
        case 3:
        AlgSelected = RAO1
        case 4:
        AlgSelected = RAO2
        case 5:
        AlgSelected = RAO3
        case 6:
        AlgSelected = TLBO
        end switch
        for k = 1 to numDesignVars do
            Compute newPop_m^k using AlgSelected
        end for
    end for
end for
Search for the current BestPop
```

Algorithm 10 shows the third proposed hybrid algorithm, dubbed HYBIND, in which a different algorithm in each iteration handles each individual of the population.

Algorithm 10 HYBIND: Hybrid algorithm based on individuals

```
NumOfAlgorithms = 7
Selection = 0
for iterator = 1 to max_ITs do
    Selection = iterator%7
    for m = 0 to popSize do
        switch (Selection)
        case 0:
        AlgSelected = Jaya
        case 1:
        AlgSelected = Chaotic Jaya
        case 2:
        AlgSelected = SCA
        case 3:
        AlgSelected = RAO1
        case 4:
        AlgSelected = RAO2
        case 5:
        AlgSelected = RAO3
        case 6:
        AlgSelected = TLBO
        end switch
        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. But, 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 Tables 1 and 2). 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.

Table 1. Benchmark functions. Names and parameters.

Id.	Name	Dim. (V)	Domain (Min,Max)
F1	Sphere	30	$-100, 100$
F2	SumSquares	30	$-10, 10$
F3	Beale	2	$-4.5, 4.5$
F4	Easom	2	$-100, 100$
F5	Matyas	2	$-10, 10$
F6	Colville	4	$-10, 10$
F7	Trid 6	6	$-V^2, V^2$
F8	Trid 10	10	$-V^2, V^2$
F9	Zakharov	10	$-5, 10$
F10	Schwefel_1.2	30	$-100, 100$

F11	Rosenbrock	30	$-30, 30$
F12	Dixon-Price	5	$-10, 10$
F13	Foxholes	2	$-2^{16}, 2^{16}$
F14	Branin	2	$x_1 : -5, 10; x_2 : 0, 15$
F15	Bohachevsky_1	2	$-100, 100$
F16	Booth	2	$-10, 10$
F17	Michalewicz_2	2	$0, \pi$
F18	Michalewicz_5	5	$0, \pi$
F19	Bohachevsky_2	2	$-100, 100$
F20	Bohachevsky_3	2	$-100, 100$
F21	GoldStein-Price	2	$-2, 2$
F22	Perm	4	$-V, V$
F23	Hartman_3	3	$0, 1$
F24	Ackley	30	$-32, 32$
F25	Penalized_2	30	$-50, 50$
F26	Langermann_2	2	$0, 10$
F27	Langermann_5	5	$0, 10$
F28	Fletcher-Powell_5	5	$x_i, \alpha_i : -\pi, \pi; a_{ij}, b_{ij} : -100, 100$

Table 2. Benchmark functions. Formulations.

Id.	Function
F1	$f = \sum_{i=1}^{V} x_i^2$
F2	$f = \sum_{i=1}^{V} i x_i^2$
F3	$f = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2$ $+(2.625 - x_1 + x_1 x_2^3)^2$
F4	$f = -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2 - (x_2-\pi)^2\right)$
F5	$f = 0.26(x_1^2 + x_2^2) - 0.48 x_1 x_2$
F6	$f = 100(x_1^2 - x_2)^2 + (x_1-1)^2 + (x_3-1)^2 + 90(x_3^2 - x_4)^2$ $+10.1\left((x_2-1)^2 + (x_4-1)^2\right) + 19.8(x_2-1)(x_4-1)$
F7 F8	$f = \sum_{i=1}^{V}(x_i - 1)^2 - \sum_{i=2}^{V} x_i x_{i-1}$
F9	$f = \sum_{i=1}^{V} x_i^2 + \left(\sum_{i=1}^{V} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{V} 0.5 i x_i\right)^4$
F10	$f = \sum_{i=1}^{V}\left(\sum_{j=1}^{i} x_j\right)^2$
F11	$f = \sum_{i=1}^{V-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$
F12	$f = (x_1 - 1)^2 + \sum_{i=2}^{V} i\left(2x_i^2 - x_{i-1}\right)^2$
F13	$f = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$
F14	$f = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$
F15	$f = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$

F16 $f = (x_1 - 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$

F17, F18 $f = -\sum_{i=1}^{V} \sin x_i \left(\sin \left(\frac{i x_i^2}{\pi} \right) \right)^{20}$

F19 $f = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$

F20 $f = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$

F21 $f = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$
$\left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$

F22 $f = \sum_{j=1}^{V} \left[\sum_{i=1}^{i} (i^j + \beta) \left(\left(\frac{x_i}{i} \right)^j - 1 \right) \right]^2$

F23 $f = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2 \right]$

F24 $f = -20\exp\left(-0.2\sqrt{\frac{1}{V}\sum_{i=1}^{V} x_i^2} \right) - \exp\left(\frac{1}{V}\sum_{i=1}^{V} \cos(2\pi x_i) \right) + 20 + e$

F25 $f = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{V-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right]$
$+ (x_V - 1)^2 \left[1 + \sin^2(2\pi x_V)\right]\} + \sum_{i=1}^{V} u(x_i, 5, 100, 4),$
$u(x_i, a, k, m) =$
$k(x_i - a)^m, x_i > a; 0, -a \le x_i \le a; k(-x_i - a)^m, x_i < -a.$

F26, F27 $f = -\sum_{i=1}^{5} c_i \left[\exp\left(-\frac{1}{\pi} \sum_{j=1}^{V} (x_j - a_{ij})^2 \right) \cos\left(\pi \sum_{j=1}^{V} (x_j - a_{ij})^2 \right) \right]$

F28 $f = \sum_{i=1}^{V} (A_i - B_i)^2;$
$A_i = \sum_{j=1}^{V} \left(a_{ij} \sin\alpha_j + bij \cos\alpha_j \right),$

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 24 of Algoritm 7), 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 ($max\_ITs = 50000$) and two population sizes ($popSize = 140$ and 210). The maximum values of the analyzed data are listed in Table 3.

Table 3. Maximum values of the analyzed data.

Population size	NoR-AI	NoR-BI	NoR-BwT	LtI-AI	LtI-BI
140	7000000	7000000	7000000	49999	49999
210	10500000	10500000	10500000	49999	49999

Tables 4-6 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. Also, 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 [80,81]). 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.

Table 4. 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	293516	3676	3434	49999	49971	75435	1580	1525	1168	1157
F2	294070	3653	3435	49999	49990	75324	1581	1526	1163	1151
F3	8957	70	60	862	778	2394	20	9	49719	28810
F4	5045	43	24	451	412	2453	26	8	49773	27786
F5	96251	739	729	8351	8236	1930	14	8	41	8
F6	17568	151	112	25293	22919	3264	54	0	49843	41426
F7	11683	115	56	49044	10094	3856	60	0	49722	37363
F8	18075	197	79	49358	28600	6030	114	0	49722	35566
F9	249027	2288	2206	49999	49968	27271	443	421	552	446
F10	7522	70	0	49993	48950	76978	1465	1411	1597	1552
F11	59193	956	375	49992	49660	9367	197	0	49925	43912
F12	7648	74	27	41236	25124	3519	56	0	49765	34756
F13	2658	23	0	49779	30524	3220	37	0	49666	38222
F14	3261	29	0	21719	17234	2023	21	0	49713	32070
F15	6084	42	22	275	234	1639	21	8	30	11
F16	2827	24	8	49678	38126	2413	27	9	49709	34212
F17	4945	36	1	4098	174	2134	18	1	49739	34560
F18	9958	96	0	48344	8329	2162	20	0	49648	36491
F19	6247	46	25	329	273	1708	19	6	32	10
F20	6258	48	24	657	444	1597	16	4	31	7
F21	2599	18	4	49818	37023	2543	28	14	49651	27864
F22	1841	15	0	48784	18724	1907	17	0	49850	24366
F23	5976	51	0	394	180	2170	19	0	49662	23570
F24	41575	415	205	9044	4874	6509	117	55	124	95
F25	58079	840	621	8652	8613	5559	75	0	49810	40191
F26	5063	43	0	13854	11630	2389	22	0	49848	32604
F27	9071	79	23	25536	9665	2127	21	0	49698	31794
F28	2119	15	0	49427	24846	2156	21	0	49635	33067

Table 5. Analysis of SCA and 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	209388	1397	1339	12387	4261	282204	4109	3847	49999	49985
F2	208824	1397	1344	12129	4222	282675	4105	3867	49999	49990
F3	3263	24	11	49999	49908	10490	71	59	1037	937
F4	3522	30	10	49999	49983	5921	41	23	819	588
F5	81841	631	622	4113	2540	102605	739	728	6077	5996
F6	6613	42	0	49999	49997	20964	154	111	45874	41787
F7	7416	60	0	49999	49999	14501	114	57	48669	19652
F8	11411	90	0	49999	49998	21868	190	77	49782	23351
F9	137414	1134	1098	13643	7062	394478	3436	3350	49992	49865
F10	131216	1416	1352	20314	13659	50688	573	299	49999	49912
F11	22671	173	0	49999	49998	49031	730	116	49986	49824
F12	7116	59	8	49999	49996	18005	144	97	15151	9236
F13	4083	37	0	49999	49996	1174	9	0	9811	8799
F14	3161	22	0	49999	49985	2342	18	0	45878	28668
F15	5976	43	22	200	66	6468	46	24	263	217
F16	3314	27	10	49999	49998	10022	70	53	352	298
F17	3030	24	0	49999	49994	5717	39	1	4279	221
F18	5461	45	0	49999	49998	12198	100	0	47566	7285
F19	6201	41	22	259	68	6648	45	25	318	268
F20	5751	42	23	493	101	6677	42	25	10328	389
F21	3330	26	12	49999	48236	6390	48	31	41485	22033
F22	2986	25	0	49999	49975	1931	13	0	49470	24078
F23	3699	29	0	49999	49951	6505	53	0	670	215
F24	22109	128	62	26470	4247	45468	441	217	8818	4045
F25	23889	206	0	49999	49998	59024	843	618	10895	10755
F26	3205	21	0	49999	49958	5850	41	0	18757	17964
F27	5376	47	0	49999	49997	6280	47	10	32659	14141
F28	5969	42	0	49999	49997	4077	33	0	45602	30802

Table 6. Analysis of RAO2, RAO3 and 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	158207	1833	1582	49999	49975	402040	6191	5990	12845	12588	317548	6636	6432	1816	1806
F2	158651	1838	1600	49999	49964	400507	6179	5990	12748	12545	314216	6582	6388	1802	1792
F3	9855	74	59	696	610	9933	74	60	665	583	10098	72	60	296	166
F4	5435	44	25	2038	2015	5509	44	25	1088	1063	5455	48	25	127	83
F5	107111	736	727	8885	8769	115700	756	744	12296	12161	151056	764	751	1261	1229
F6	16646	129	93	49995	49669	16154	127	86	49984	49810	46797	185	136	5463	3573
F7	11737	108	56	48902	7854	11421	115	61	48508	13021	13459	139	70	49638	5639
F8	17869	218	63	49543	19788	17255	188	76	48896	25615	202435	3127	1418	49440	24482
F9	158199	1411	1324	49999	49965	209334	2857	2785	17784	17582	277040	3261	3180	2985	2958
F10	3889	31	0	49943	46966	58363	716	536	49998	49889	409579	5276	5097	8337	8270
F11	34536	524	8	49940	49106	47480	905	7	49984	49761	4325394	65328	26532	42857	38436
F12	19104	187	136	1472	1393	12595	123	57	6960	3570	25139	234	172	393	299
F13	640	9	0	49680	21886	1285	14	0	49649	30857	5251	49	0	48857	1302
F14	4472	30	0	4716	1414	4423	32	0	6115	3465	4862	33	0	2518	64
F15	6939	43	22	667	348	6474	43	23	319	275	8420	48	26	63	51
F16	9812	69	53	534	464	2904	22	7	49838	32614	11723	69	54	103	86
F17	5089	40	1	2641	149	5103	40	1	2187	144	5888	35	1	2137	56
F18	9466	95	0	48411	1938	9493	93	0	49151	2102	14358	122	0	49208	1466
F19	7104	49	27	833	547	6677	46	24	561	509	7812	43	21	73	58
F20	6847	42	23	3782	1677	5730	45	26	1115	1044	8677	47	25	103	75
F21	6889	45	28	49200	10483	2627	21	4	49803	33314	6407	46	30	27592	9567
F22	2050	15	0	47791	22625	2381	23	0	49441	35495	27875	218	55	49997	47112
F23	6113	49	0	241	151	6095	52	0	249	147	7185	57	0	149	57
F24	38389	371	182	32501	15916	27517	355	177	1532	841	25248	402	201	237	167
F25	52014	758	554	4811	4799	49682	741	545	1584	1570	140740	2360	837	21162	7098
F26	5153	39	0	1061	883	5263	39	0	6302	1436	6689	38	0	15868	758
F27	9039	93	16	14656	4382	9148	92	23	19748	2674	12647	222	124	21151	5807
F28	3993	33	0	43661	35225	2318	18	0	49411	26332	38285	238	162	19500	897

An important aspect, not shown in Tables 4-6, is whether the solution obtained by each algorithm is acceptable or not. Table 7 shows whether the obtained solution tolerance is less than 0.001 (Y) or not (N). As seen from this table, 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.

Table 7. Quality of solutions for the considered algorithms with a tolerance of 0.001 and a population of 140.

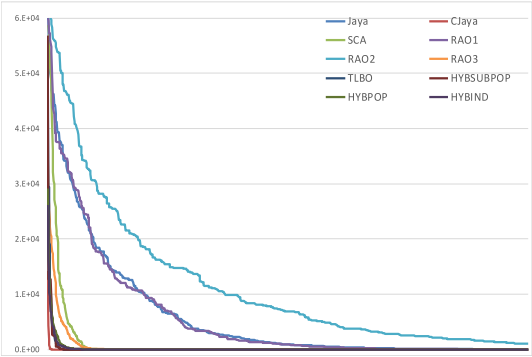
	Jaya	Chaotic Jaya	SCA	RAO1	RAO2	RAO3	TLBO
F1	Y	Y	Y	Y	Y	Y	Y
F2	Y	Y	Y	Y	Y	Y	Y
F3	Y	Y	Y	Y	Y	Y	Y
F4	Y	Y	Y	Y	Y	Y	Y
F5	Y	Y	Y	Y	Y	Y	Y
F6	Y	Y	Y	Y	Y	Y	Y
F7	Y	Y	Y	Y	Y	Y	Y
F8	Y	N	Y	Y	N	Y	Y
F9	Y	Y	Y	Y	Y	Y	Y
F10	N	Y	Y	Y	N	Y	Y
F11	Y	N	N	Y	Y	Y	Y
F12	Y	N	Y	Y	Y	N	Y
F13	Y	N	Y	N	N	N	Y
F14	Y	Y	Y	Y	Y	Y	Y
F15	Y	Y	Y	Y	Y	Y	Y
F16	Y	Y	Y	Y	Y	Y	Y
F17	Y	Y	Y	Y	Y	Y	Y
F18	Y	N	Y	Y	N	N	N
F19	Y	Y	Y	Y	Y	Y	Y
F20	Y	Y	Y	Y	Y	Y	Y
F21	Y	Y	Y	Y	Y	Y	Y
F22	Y	Y	Y	Y	Y	Y	Y
F23	Y	Y	Y	N	Y	Y	Y
F24	Y	Y	Y	Y	N	Y	Y
F25	Y	N	Y	Y	Y	Y	N
F26	Y	Y	Y	Y	Y	Y	Y
F27	N	N	N	N	N	N	Y
F28	N	N	Y	N	N	N	Y

Table 8 compares the quality of the solutions obtained from the proposed hybrid algorithms. It is clear 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. Moreover, the HYBIND algorithm has a slightly better performance in comparison to HYBPOP.

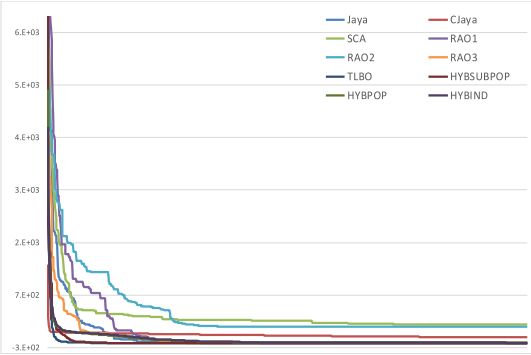
Table 8. Quality of solutions for the hybrid algorithms with a tolerance of 0.001.

	popSize=140			popSize=210		
	HYBSUBPOP	HYBPOP	HYBIND	HYBSUBPOP	HYBPOP	HYBIND
F1	Y	Y	Y	Y	Y	Y
F2	Y	Y	Y	Y	Y	Y
F3	Y	Y	Y	Y	Y	Y
F4	Y	Y	Y	Y	Y	Y
F5	Y	Y	Y	Y	Y	Y
F6	Y	Y	Y	Y	Y	Y
F7	Y	Y	Y	Y	Y	Y
F8	Y	Y	Y	Y	Y	Y
F9	Y	Y	Y	Y	Y	Y
F10	Y	Y	Y	Y	Y	Y
F11	N	Y	N	N	Y	Y
F12	Y	Y	Y	Y	Y	Y
F13	Y	Y	Y	Y	Y	Y
F14	Y	Y	Y	Y	Y	Y
F15	Y	Y	Y	Y	Y	Y
F16	Y	Y	Y	Y	Y	Y
F17	Y	Y	Y	Y	Y	Y
F18	Y	Y	Y	Y	Y	Y
F19	Y	Y	Y	Y	Y	Y
F20	Y	Y	Y	Y	Y	Y
F21	Y	Y	Y	Y	Y	Y
F22	Y	Y	Y	Y	Y	Y
F23	N	Y	Y	N	Y	Y
F24	Y	Y	Y	Y	Y	Y
F25	Y	Y	Y	Y	Y	Y
F26	Y	Y	Y	Y	Y	Y
F27	N	N	Y	N	N	Y
F28	Y	Y	Y	Y	Y	Y

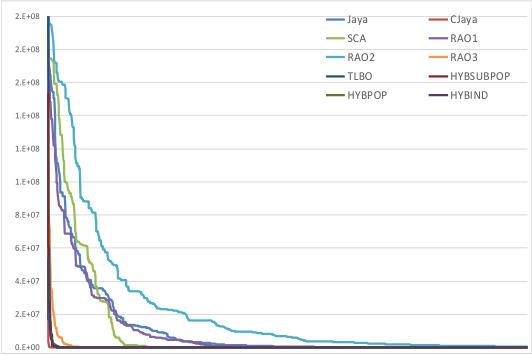
Local exploration has improved both in the HYBPOP method and especially in the HYBIND method, as shown in Tables 7 and 8. Figures 1 and 2 show the convergence curves of both all the individual methods and the three hybrid methods proposed for the first 1000 and 100 iterations respectively, for functions F1, F8, F11 and F18. Each point in both figures is the average of the data obtained in 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.



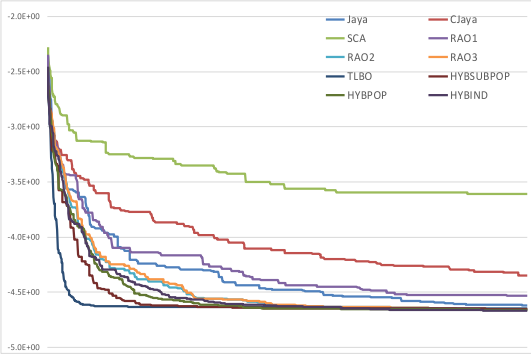
(a) F1



(b) F8

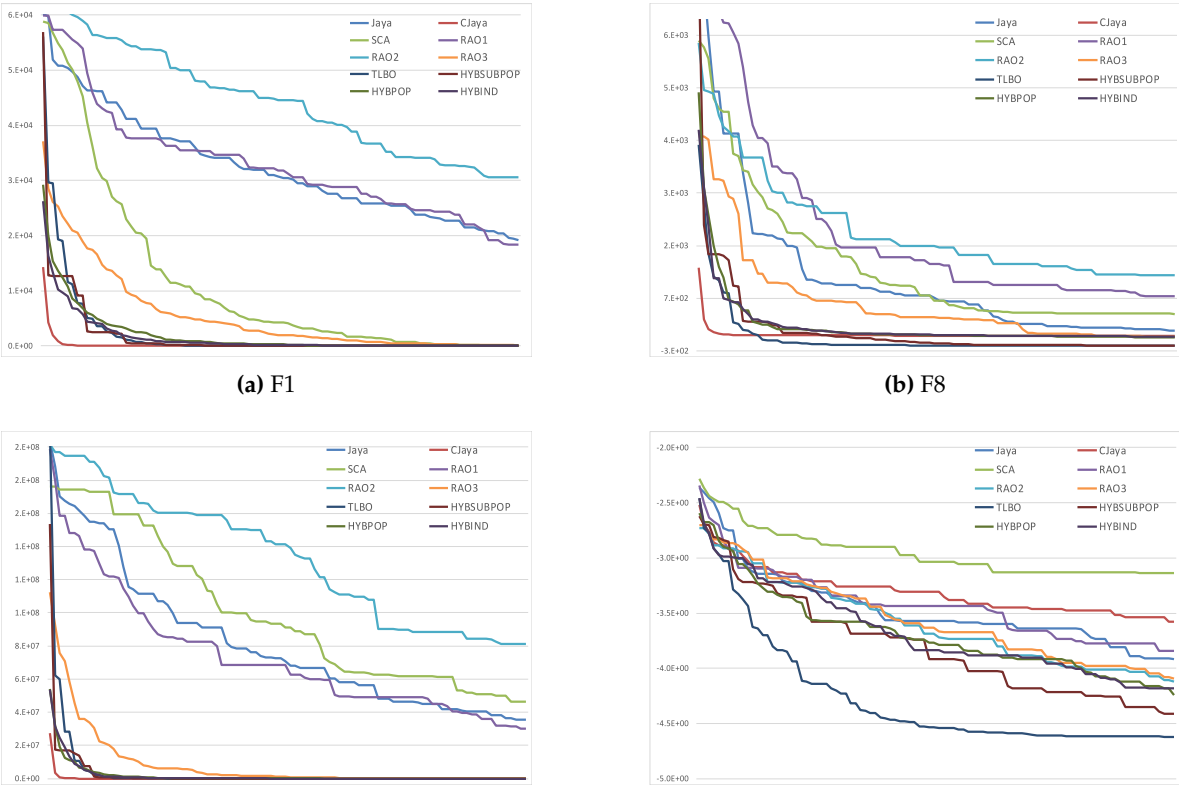


(c) F11



(d) F18

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



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

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

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

An analysis of the contribution of each algorithm in the HYBPOP and HYBIND algorithms is exhibited in Tables 9-11. Table 9 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 9, the HYBIND algorithm performs more replacements of individuals. Also, 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.

Table 9. Contribution of each algorithm to the replacements of the individuals (NoR-AI). The population size is set as 140.

	HYBPOP							HYBIND						
	Jaya	Chaotic Jaya	SCA	RAO1	RAO2	RAO3	TLBO	Jaya	Chaotic Jaya	SCA	RAO1	RAO2	RAO3	TLBO
F1	25125	14555	96302	7	17725	332520	111730	41978	16719	17985	6	41958	12560	16529
F2	25246	12585	96155	9	17679	333194	98347	41855	16147	18153	6	41851	12747	16037
F3	2873	156	182	323	2594	6666	6861	49975	36436	48752	171	49885	49959	1642
F4	1368	232	511	21	1374	2348	3590	49892	27089	49998	53	49917	49998	570
F5	4483	1028	39631	33	4705	52409	1444	7280	267	6607	25	7281	4726	2744
F6	1137	285	259	89	780	146009	31692	49059	48335	49999	1034	49763	49816	47269
F7	2909	253	135	873	3851	5720	5898	49995	43638	49999	59	49996	49998	46348
F8	3570	369	180	840	4777	13342	15701	49990	45549	49999	16	49993	49997	49803
F9	9658	7123	64165	116	2912	216548	23519	49998	4623	18878	65	49998	11271	6159
F10	1055	13634	27550	51	198	535232	109601	49987	29386	49999	45	49946	49780	29077
F11	4617	536	834	818	3734	124739	301758	49683	24945	49999	7	49608	46959	49999
F12	3825	393	421	488	5401	6225	19531	49072	43353	49999	443	49822	49922	8679
F13	43	409	1617	2	1530	8015	4041	16920	49248	49999	703	49922	49998	39259
F14	1100	91	239	114	985	542	3217	49507	43442	49999	359	49513	49870	3924
F15	670	926	2581	38	651	3111	1274	308	167	295	28	307	285	184
F16	272	142	155	120	4817	10285	6404	25143	48481	49996	64	49897	49970	888
F17	1614	80	62	504	1609	2146	3008	49989	30269	49998	75	49987	49998	16011
F18	2931	85	64	222	4076	4234	10162	49367	45600	49999	736	49354	49679	26117
F19	512	975	3020	38	489	2550	1368	479	174	447	37	474	435	192
F20	594	985	2289	38	531	4788	1270	708	210	633	52	694	623	245
F21	164	189	302	209	2708	8114	3782	8538	49320	49767	51	49443	49940	47027
F22	380	66	558	240	378	866	5278	47536	44069	49999	18746	46949	47932	49985
F23	1885	62	45	689	2101	2698	3214	49995	39615	49999	58	49991	49998	1276
F24	6959	1882	7941	27	5402	50267	7073	17557	1549	24593	3479	12641	9469	43818
F25	6308	368	164	2774	21124	28995	29346	49942	37071	49999	10730	49992	49990	43810
F26	1524	122	306	146	1496	1971	3772	46010	45348	49999	1303	47091	49816	13990
F27	1488	77	869	311	1909	3090	7708	43480	44977	49999	9953	47560	46582	27162
F28	184	100	1674	210	875	1065	27356	23923	48894	49999	2436	43178	45987	45473

Table 10 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 9, 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.

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

	HYBPOP							HYBIND						
	Jaya	Chaotic Jaya	SCA	RAO1	RAO2	RAO3	TLBO	Jaya	Chaotic Jaya	SCA	RAO1	RAO2	RAO3	TLBO
F1	27706	15952	27672	8	27687	27638	15974	41978	16719	17985	6	41958	12560	16529
F2	27598	14205	27566	7	27585	27535	14220	41855	16147	18153	6	41851	12747	16037
F3	1115	158	192	1077	1115	1091	1092	49975	36436	48752	171	49885	49959	1642
F4	668	138	485	138	671	660	533	49892	27089	49998	53	49917	49998	570
F5	8136	301	8124	24	8146	8100	297	7280	267	6607	25	7281	4726	2744
F6	49889	1032	853	4730	49915	49740	46411	49059	48335	49999	1034	49763	49816	47269
F7	42814	120	80	41573	45096	45055	40202	49995	43638	49999	59	49996	49998	46348
F8	48715	320	256	46062	47122	48564	49177	49990	45549	49999	16	49993	49997	49803
F9	30943	4732	30905	67	30884	30842	4733	49998	4623	18878	65	49998	11271	6159
F10	49434	27375	49999	44	46704	49999	27435	49987	29386	49999	45	49946	49780	29077
F11	14618	2561	40446	13232	14525	49776	49999	49683	24945	49999	7	49608	46959	49999
F12	9357	302	5310	5055	10953	14699	9658	49072	43353	49999	443	49822	49922	8679
F13	14277	1063	14850	1830	46409	47589	46804	16920	49248	49999	703	49922	49998	39259
F14	30447	174	30620	627	30428	27404	2988	49507	43442	49999	359	49513	49870	3924
F15	303	181	302	37	303	297	183	308	167	295	28	307	285	184
F16	267	82	110	157	1631	1527	863	25143	48481	49996	64	49897	49970	888
F17	4402	72	64	2665	4278	3223	3444	49989	30269	49998	75	49987	49998	16011
F18	44394	6650	288	43185	48625	46910	46650	49367	45600	49999	736	49354	49679	26117
F19	385	195	375	30	380	359	194	479	174	447	37	474	435	192
F20	663	211	640	39	653	615	209	708	210	633	52	694	623	245
F21	425	158	529	41474	48520	47210	45157	8538	49320	49767	51	49443	49940	47027
F22	44939	15050	49999	25381	46460	44796	49990	47536	44069	49999	18746	46949	47932	49985
F23	443	59	34	355	472	421	435	49995	39615	49999	58	49991	49998	1276
F24	22208	1723	25856	11	13680	4089	1431	17557	1549	24593	3479	12641	9469	43818
F25	12972	499	1683	8051	13175	11759	12305	49942	37071	49999	10730	49992	49990	43810
F26	4182	2569	3561	2341	4211	4082	3674	46010	45348	49999	1303	47091	49816	13990
F27	26951	344	25400	9212	23800	30524	31150	43480	44977	49999	9953	47560	46582	27162
F28	20387	641	49999	7204	45865	47674	42103	23923	48894	49999	2436	43178	45987	45473

Finally, Tables 11 and 12 show the last iteration in which each algorithm obtains a new optimum and the total number of replacements for the current best individual, respectively. A careful analysis of the results in Table 11 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, as shown in Table 12. 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.

Table 11. Last iteration in which a replacement of the best individual occurs (LtI-BI). The population size is set as 140.

	HYBPOP							HYBIND						
	Jaya	Chaotic Jaya	SCA	RAO1	RAO2	RAO3	TLBO	Jaya	Chaotic Jaya	SCA	RAO1	RAO2	RAO3	TLBO
F1	0	15863	0	0	0	0	15730	0	15946	14473	0	0	0	15509
F2	0	12473	0	0	0	0	12344	0	15348	12873	0	0	0	15188
F3	475	33	2	408	329	765	943	6	16	7	2	0	31	1288
F4	1	48	19	7	0	16	409	0	27	27	4	0	26	450
F5	0	38	10	0	0	0	17	0	68	21	0	0	4	8
F6	24	96	6	3	21	6850	32134	4	15	72	39	0	85	35901
F7	1764	23	11	1941	1077	5315	5811	0	21	2	10	2	46	24451
F8	5681	78	0	8600	8735	13766	18316	0	45	0	0	0	193	44652
F9	0	3322	0	0	0	1	3241	0	3308	0	3	0	1	3086
F10	0	25452	0	0	0	1	25355	0	27404	7926	0	0	0	27330
F11	10586	1241	0	12384	8518	29734	49977	0	357	0	0	0	0	49996
F12	750	84	22	867	741	758	2766	17	125	16	84	20	250	4035
F13	1	330	71	1	1	3	8260	1	476	374	2	1	2	14327
F14	11	5	0	1	1	5	414	0	5	2	1	1	6	560
F15	0	62	10	0	0	0	25	0	41	22	0	0	2	6
F16	1	29	4	23	9	19	705	1	14	2	8	1	23	753
F17	168	5	0	154	114	195	239	2	1	0	0	1	6	645
F18	6493	6472	5	6667	8367	7576	8274	3	2	0	534	8	3	14666
F19	1	59	8	0	0	2	11	0	66	51	0	0	6	12
F20	0	49	1	0	0	1	37	0	61	34	0	0	0	37
F21	1	47	8	8820	6337	9098	5499	1	32	10	2	0	4850	16989
F22	88	276	48	55	19	197	44183	3	0	0	74	39	99	46093
F23	211	5	0	177	151	213	247	1	8	0	6	1	3	564
F24	0	1034	0	0	0	0	786	0	1144	0	0	0	0	757
F25	3665	232	0	3948	3720	3687	3792	0	116	4662	10704	0	0	43752
F26	26	59	2	6	9	5	632	11	59	9	11	10	20	900
F27	501	16	5004	328	350	2802	13485	4058	78	20001	27	253	2746	11151
F28	2	36	2	18	4	9	11474	0	13	3	30	11	16	11993

Table 12. Total number of replacements for the current best individual (NoR-BI). The population size is set as 140.

	HYBPOP							NoR-AI						
	Jaya	Chaotic Jaya	SCA	RAO 1	RAO 2	RAO 3	TLBO	Jaya	Chaotic Jaya	SCA	RAO 1	RAO 2	RAO 3	TLBO
F1	0	1402	0	0	0	0	132	0	1196	63	0	0	0	111
F2	0	1098	0	0	0	0	105	0	1172	24	0	0	0	120
F3	2	2	1	2	1	7	64	0	2	1	0	0	2	72
F4	0	4	2	1	0	1	34	0	4	1	0	0	2	38
F5	0	8	1	0	0	0	2	0	14	2	0	0	1	1
F6	0	6	0	0	0	15	170	0	4	1	1	0	1	280
F7	6	5	0	7	4	13	83	0	6	0	0	0	2	349
F8	15	14	0	13	9	37	149	0	12	0	0	0	1	3305
F9	0	280	0	0	0	1	62	0	268	0	1	0	1	24
F10	0	915	0	0	0	0	574	0	949	3	0	0	0	438
F11	17	38	0	30	13	246	3020	0	32	0	0	0	0	6206
F12	5	13	0	4	2	7	200	0	13	0	0	0	4	332
F13	0	10	3	0	0	1	33	0	8	6	0	0	0	34
F14	0	1	0	0	0	1	27	0	1	1	0	0	1	30
F15	0	16	1	0	0	0	1	0	10	3	0	0	1	1
F16	0	2	0	1	1	2	66	0	2	1	0	0	3	66
F17	2	1	0	2	1	6	23	0	1	0	0	0	1	36
F18	1	1	0	3	2	3	116	0	0	0	1	0	1	160
F19	0	14	1	0	0	0	1	0	13	7	0	0	0	1
F20	0	15	1	0	0	0	3	0	12	4	0	0	0	1
F21	0	3	1	2	1	2	40	0	3	1	0	0	2	41
F22	1	0	1	1	1	3	51	0	0	0	1	1	3	158
F23	5	1	0	4	3	7	35	1	1	0	0	1	1	51
F24	0	115	0	0	0	0	11	0	112	0	0	0	0	6
F25	9	20	0	33	48	35	739	0	17	0	0	0	0	4484
F26	1	1	0	1	1	1	33	1	1	0	1	1	1	34
F27	2	1	2	2	2	4	116	1	2	8	1	1	2	172
F28	1	1	1	2	1	1	245	0	0	1	2	0	2	490

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 a future line of work, we intend to integrate more efficient algorithms in the proposed hybridization skeleton and extend the performance analysis of the potential algorithms for solving real-world engineering problems.

Author Contributions: Héctor Migallón, Akram Belazi, José-Luis Sánchez Romero, Héctor Rico and Antonio Jimeno-Morenilla conceived the hybrid algorithms; Akram Belazi conceived the Chaotic 2D Jaya algorithm; Héctor Migallón designed the hybrid algorithms; Héctor Migallón codified the hybrid algorithms; Héctor Migallón, José-Luis Sánchez Romero and Héctor Rico performed numerical experiments; Héctor Migallón, Akram Belazi and Antonio Jimeno-Morenilla analyzed the data; Héctor Migallón wrote the original draft. Akram Belazi, José-Luis Sánchez Romero, Héctor Rico and Antonio Jimeno-Morenilla reviewed and edited the manuscript. All the 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. *Applied Soft Computing* **2013**, *13*, 2592–2612. doi:10.1016/j.asoc.2012.11.026.

2. Zhao, W.; Zhang, Z.; Wang, L. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Engineering Applications of Artificial Intelligence* **2020**, *87*, 103300. doi:10.1016/j.engappai.2019.103300.
3. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers and Structures* **2016**, *169*, 1 – 12. doi:10.1016/j.compstruc.2016.03.001.
4. Dorigo, M.; Di Caro, G. New Ideas in Optimization; McGraw-Hill Ltd., UK: Maidenhead, UK, England, 1999; chapter The Ant Colony Optimization Meta-heuristic, pp. 11–32.
5. Ma, H.; Simon, D.; Siarry, P.; Yang, Z.; Fei, M. Biogeography-Based Optimization: A 10-Year Review. *IEEE Transactions on Emerging Topics in Computational Intelligence* **2017**, *1*, 391–407. doi:10.1109/TETCI.2017.2739124.
6. Ahrari, A.; Atai, A.A. Grenade Explosion Method—A novel tool for optimization of multimodal functions. *Applied Soft Computing* **2010**, *10*, 1132 – 1140. doi:10.1016/j.asoc.2009.11.032.
7. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intelligence* **2007**, *1*, 33–57. doi:10.1007/s11721-007-0002-0.
8. Xin-She, Y. Firefly Algorithm, Lévy Flights and Global Optimization. *Research and Development in Intelligent Systems XXVI* **2009**, *á*, 209–218. doi:10.1007/978-1-84882-983-1_15.
9. Karaboga, D.; Basturk, B. On the Performance of Artificial Bee Colony (ABC) Algorithm. *Appl. Soft Comput.* **2008**, *8*, 687–697. doi:10.1016/j.asoc.2007.05.007.
10. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Information Sciences* **2009**, *179*, 2232 – 2248. Special Section on High Order Fuzzy Sets, doi:10.1016/j.ins.2009.03.004.
11. Eusuff, M.; Lansey, K.; Pasha, F. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization* **2006**, *38*, 129–154. doi:10.1080/03052150500384759.
12. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems* **2016**, *96*, 120 – 133. doi:10.1016/j.knosys.2015.12.022.
13. Rao, R.V.; Savsani, V.; Vakharia, D. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design* **2011**, *43*, 303–315. doi:10.1016/j.cad.2010.12.015.
14. Zhao, W.; Wang, L.; Zhang, Z. Supply-Demand-Based Optimization: A Novel Economics-Inspired Algorithm for Global Optimization. *IEEE Access* **2019**, *7*, 73182–73206. doi:10.1109/ACCESS.2019.2918753.
15. Rao, R.V. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations* **2016**, *7*, 19–34. doi:10.5267/j.ijiec.2015.8.004.
16. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems* **2019**, *97*, 849 – 872. doi:10.1016/j.future.2019.02.028.
17. Rao, R.V. Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems. *International Journal of Industrial Engineering Computations* **2020**, *11*, 107 – 130. doi:10.5267/j.ijiec.2019.6.002.
18. Majumdar, M.; Mitra, T.; Nishimura, K. *Optimization and Chaos*; Springer, New York, 2000.
19. Ott, E., Frontmatter. In *Chaos in Dynamical Systems*, 2 ed.; Cambridge University Press, 2002; pp. i–iv.
20. Gandomi, A.; Yang, X.S.; Talatahari, S.; Alavi, A. Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation* **2013**, *18*, 89 – 98. doi:10.1016/j.cnsns.2012.06.009.
21. Gokhale, S.; Kale, V. An application of a tent map initiated Chaotic Firefly algorithm for optimal overcurrent relay coordination. *International Journal of Electrical Power & Energy Systems* **2016**, *78*, 336 – 342. doi:10.1016/j.ijepes.2015.11.087.
22. Ma, Z.S. Chaotic populations in genetic algorithms. *Applied Soft Computing* **2012**, *12*, 2409 – 2424. doi:10.1016/j.asoc.2012.03.001.
23. Yan, X.F.; Chen, D.Z.; Hu, S.X. Chaos-genetic algorithms for optimizing the operating conditions based on RBF-PLS model. *Computers & Chemical Engineering* **2003**, *27*, 1393 – 1404. doi:10.1016/S0098-1354(03)00074-7.
24. Hong, W.C. Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm. *Neurocomputing* **2011**, *74*, 2096 – 2107. doi:10.1016/j.neucom.2010.12.032.
25. Mingjun, J.; Huanwen, T. Application of chaos in simulated annealing. *Chaos, Solitons & Fractals* **2004**, *21*, 933 – 941. doi:10.1016/j.chaos.2003.12.032.

26. Saremi, J.; Mirjalili, S.; Lewisn, A. Biogeography-based optimisation with chaos. *Neural Computing and Applications* **2014**, *25*, 1077 – 1097. doi:10.1007/s00521-014-1597-x.
27. Wang, X.; Duan, H. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Computers & Industrial Engineering* **2014**, *73*, 96 – 114. doi:10.1016/j.cie.2014.04.006.
28. Jia, D.; Zheng, G.; Khan, M.K. An effective memetic differential evolution algorithm based on chaotic local search. *Information Sciences* **2011**, *181*, 3175 – 3187. doi:10.1016/j.ins.2011.03.018.
29. 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. *International Journal of Electrical Power & Energy Systems* **2012**, *42*, 119 – 126. doi:10.1016/j.ijepes.2012.03.012.
30. Alatas, B. Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications* **2010**, *37*, 5682 – 5687. doi:10.1016/j.eswa.2010.02.042.
31. 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*. doi:10.3390/en11081946.
32. Farah, A.; Belazi, A. A novel chaotic Jaya algorithm for unconstrained numerical optimization. *Nonlinear Dynamics* **2018**, *93*, 1451 – 1480. doi:10.1007/s11071-018-4271-5.
33. Kumar, Y.; Singh, P.K. RA chaotic teaching learning based optimization algorithm for clustering problems. *Applied Intelligence* **2019**, *49*, 107 – 130. doi:10.1007/s10489-018-1301-4.
34. 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. doi:10.1016/j.measurement.2020.107576.
35. 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. doi:10.1109/ACCESS.2020.2988717.
36. 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. *Applied Sciences* **2020**, *10*. doi:10.3390/app10072301.
37. 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. *Computational Intelligence* **2020**, *36*, 132–150. doi:10.1111/coin.12247.
38. 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*. doi:10.3390/math8040546.
39. Yue, Z.; Zhang, S.; Xiao, W. A Novel Hybrid Algorithm Based on Grey Wolf Optimizer and Fireworks Algorithm. *Sensors* **2020**, *20*. doi:10.3390/s20072147.
40. 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*. doi:10.3390/su12104023.
41. Chen, X.; Yu, K. Hybridizing cuckoo search algorithm with biogeography-based optimization for estimating photovoltaic model parameters. *Solar Energy* **2019**, *180*, 192 – 206. doi:10.1016/j.solener.2019.01.025.
42. Zhang, X.; Shen, X.; Yu, Z. A Novel Hybrid Ant Colony Optimization for a Multicast Routing Problem. *Algorithms* **2019**, *12*. doi:10.3390/a12010018.
43. 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*. doi:10.3390/en12122333.
44. 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*. doi:10.3390/en12163052.
45. 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*. doi:10.3390/e21050494.
46. 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*. doi:10.3390/su11143862.

47. hanem, W.A.H.M.; Jantan, A. Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems. *Neural Computing and Applications* **2018**, *30*, 163–181. doi:10.1007/s00521-016-2665-1.
48. Das, P.; Behera, H.; Panigrahi, B. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm and Evolutionary Computation* **2016**, *28*, 14 – 28. doi:10.1016/j.swevo.2015.10.011.
49. Wang, G.G.; Gandomi, A.H.; Zhao, X.; Chu, H.C.E. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Computing* **2016**, *20*, 273–285. doi:10.1007/s00500-014-1502-7.
50. 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. *Journal of Systems Engineering and Electronics* **2015**, *26*, 317–328. doi:10.1109/JSEE.2015.00037.
51. 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*. doi:10.3390/en11061464.
52. Mishra, S.; Ray, P.K. Power quality improvement using photovoltaic fed DSTATCOM based on JAYA optimization. *IEEE Transactions on Sustainable Energy* **2016**, *7*, 1672–1680. doi:10.1109/TSTE.2016.2570256.
53. 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 Transactions on Sustainable Energy* **2018**, *9*, 45–55. doi:10.1109/TSTE.2017.2714705.
54. 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. *Engineering with Computers* **2016**, pp. 1–19. doi:10.1007/s00366-016-0484-8.
55. Choudhary, A.; Kumar, M.; Unune, D.R. Investigating effects of resistance wire heating on AISI 1023 weldment characteristics during ASAW. *Materials and Manufacturing Processes* **2018**, *33*, 759–769. doi:10.1080/10426914.2017.1415441.
56. Dinh-Cong, D.; Dang-Trung, H.; Nguyen-Thoi, T. An efficient approach for optimal sensor placement and damage identification in laminated composite structures. *Advances in Engineering Software* **2018**, *119*, 48 – 59. doi:10.1016/j.advengsoft.2018.02.005.
57. 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. *Engineering Applications of Artificial Intelligence* **2017**, *60*, 35–44. doi:10.1016/j.engappai.2017.01.008.
58. 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. *The Journal of Supercomputing* **2017**, *73*, 591–606. doi:10.1007/s11227-016-1914-5.
59. Kumar-Majhi, S. An Efficient Feed Foreword Network Model with Sine Cosine Algorithm for Breast Cancer Classification. *International Journal of System Dynamics Applications (IJSDA)* **2018**, pp. ID–202397. doi:10.4018/IJSDA.2018040101.
60. Rajesh K.S., D.S. Load frequency control of autonomous power system using adaptive fuzzy based PID controller optimized on improved sine cosine algorithm. *Journal of Ambient Intelligence and Humanized Computing* **2019**, *10*, 2361–2373. doi:10.1007/s12652-018-0834-z1.
61. 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*. doi:10.3390/en11020420.
62. Ramanaiah, M.L.; Reddy, M.D. Sine Cosine Algorithm for Loss Reduction in Distribution System with Unified Power Quality Conditioner. *i-manager's Journal on Power Systems Engineering* **2017**, *5*, 10–16. doi:10.26634/jps.5.3.13667.
63. 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. doi:10.1016/j.energy.2018.01.076.
64. Singh, V.P. Sine cosine algorithm based reduction of higher order continuous systems. 2017 International Conference on Intelligent Sustainable Systems (ICISS), 2017, pp. 649–653. doi:10.1109/ISS1.2017.8389252.
65. Das, S.; Bhattacharya, A.; Chakraborty, A.K. Solution of short-term hydrothermal scheduling using sine cosine algorithm. *Soft Computing* **2018**, *22*, 6409 – 6427. doi:10.1007/s00500-017-2695-3.

66. Rao, R.V.; Pawar, R.B. Self-adaptive Multi-population Rao Algorithms for Engineering Design Optimization. *Applied Artificial Intelligence* **2020**, *34*, 187–250. doi:10.1080/08839514.2020.1712789.
67. Rao, R.; Pawar, R. Constrained design optimization of selected mechanical system components using Rao algorithms. *Applied Soft Computing* **2020**, *89*, 106141. doi:https://doi.org/10.1016/j.asoc.2020.106141.
68. Rao, R.V.; Keesari, H.S. Rao algorithms for multi-objective optimization of selected thermodynamic cyclesn. *Engineering with Computers* **2020**. doi:10.1007/s00366-020-01008-9.
69. Singh, M.; Panigrahi, B.; Abhyankar, A. Optimal coordination of directional over-current relays using Teaching Learning-Based Optimization (TLBO) algorithm. *International Journal of Electrical Power & Energy Systems* **2013**, *50*, 33 – 41. doi:10.1016/j.ijepes.2013.02.011.
70. 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. *Engineering Applications of Artificial Intelligence* **2012**, *25*, 1577 – 1588. doi:10.1016/j.engappai.2012.07.004.
71. Li, D.; Zhang, C.; Shao, X.; Lin, W. A multi-objective TLBO algorithm for balancing two-sided assembly line with multiple constraints. *Journal of Intelligent Manufacturing* **2016**, *27*, 725–739. doi:10.1007/s10845-014-0919-2.
72. Arya, L.; Koshti, A. Anticipatory load shedding for line overload alleviation using Teaching learning based optimization (TLBO). *International Journal of Electrical Power & Energy Systems* **2014**, *63*, 862 – 877. doi:10.1016/j.ijepes.2014.06.066.
73. Mohanty, B. TLBO optimized sliding mode controller for multi-area multi-source nonlinear interconnected AGC system. *International Journal of Electrical Power & Energy Systems* **2015**, *73*, 872 – 881. doi:10.1016/j.ijepes.2015.06.013.
74. 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. doi:10.1016/j.neucom.2015.12.081.
75. 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. doi:10.1016/j.fuel.2018.05.166.
76. Migallón, H.; Jimeno-Morenilla, A.; Sánchez-Romero, J.; Belazi, A. Efficient parallel and fast convergence chaotic Jaya algorithms. *Swarm and Evolutionary Computation* **2020**, p. 100698. doi:10.1016/j.swevo.2020.100698.
77. 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*. doi:10.3390/en11081946.
78. Ravipudi, J.L.; Neebha, M. Synthesis of linear antenna arrays using Jaya, self-adaptive Jaya and chaotic Jaya algorithms. *AEU - International Journal of Electronics and Communications* **2018**, *92*, 54 – 63. doi:10.1016/j.aeue.2018.05.022.
79. Free Software Foundation, Inc.. GCC, the GNU Compiler Collection. *https://www.gnu.org/software/gcc/index.html*.
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*. doi:10.3390/electronics8010019.
81. Migallón, H.; Jimeno-Morenilla, A.; Sánchez-Romero, J.; Belazi, A. Efficient parallel and fast convergence chaotic Jaya algorithms. *Swarm and Evolutionary Computation* **2020**, p. 100698. doi:10.1016/j.swevo.2020.100698.