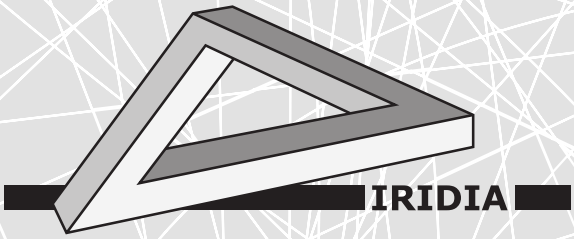

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

An Incremental $\mathbf{ACO}_{\mathbb{R}}$ with Local Search for Continuous Optimization Problems

Tianjun Liao, Marco A. Montes de Oca, Doğan Aydin, Thomas Stützle, and Marco Dorigo

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2011-005

February 2011

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2011-005

An Incremental $\mathbf{ACO}_{\mathbb{R}}$ with Local Search for Continuous Optimization Problems

Tianjun Liao† tliao@ulb.ac.be
Marco A. Montes de Oca† mmontes@ulb.ac.be
Doğan Aydin‡ dogan.aydin@ege.edu.tr
Thomas Stützle† stuetzle@ulb.ac.be
Marco Dorigo† mdorigo@ulb.ac.be

† IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium.
‡ Department of Computer Engineering, Ege University, Izmir, Turkey.

February 2011

Abstract

One of the most popular Ant Colony Optimization (ACO) algorithms for continuous optimization problems is $\text{ACO}_{\mathbb{R}}$. In this paper, we propose an incremental $\text{ACO}_{\mathbb{R}}$ with local search ($\text{IACO}_{\mathbb{R}}$-LS) that obtains better results than the original $\text{ACO}_{\mathbb{R}}$ on a number of continuous optimization problems. We first present a mechanism that improves the search diversification of $\text{ACO}_{\mathbb{R}}$. This mechanism consists of a growing solution archive with a special initialization rule applied to entrant solutions. The resulting algorithm, called $\text{IACO}_{\mathbb{R}}$, is then hybridized with a local search procedure in order to enhance its search intensification. We experiment with Powell's conjugate directions set, Powell's BOBYQA, and Lin-Yu Tseng's Mtsls1 methods as local search procedures. Automatic parameter tuning results show that $\text{IACO}_{\mathbb{R}}$-LS with Mtsls1 ($\text{IACO}_{\mathbb{R}}$-Mtsls1) is not only a significant improvement over $\text{ACO}_{\mathbb{R}}$, but that it is also competitive with the state-of-the-art algorithms described in a recent special issue of the journal Soft Computing. Further experimentation with $\text{IACO}_{\mathbb{R}}$-Mtsls1 on the combined benchmark functions suite from both the special issue of Soft Computing and the IEEE 2005 Congress on Evolutionary Computation (CEC 2005) demonstrates its good performance in continuous optimization.

1 Introduction

Several algorithms based on or inspired by the Ant Colony Optimization (ACO) metaheuristic [5] have been proposed to tackle continuous optimization problems [3, 6, 10, 11, 13, 15, 19]. One of the most popular ACO-based algorithms for continuous domains is $\text{ACO}_{\mathbb{R}}$ [22–24]. Recently, Leguizamón and Coello [12] proposed modifications that improved the performance of $\text{ACO}_{\mathbb{R}}$ on six benchmark functions. Unfortunately, the experimental results presented in [12] are far

from being competitive with the results obtained by state-of-the-art continuous optimization algorithms recently featured in a special issue of the journal Soft Computing [14] (we will refer to this special issue as SOCO in the rest of this paper). The set of algorithms described in SOCO consists of differential evolution algorithms, memetic algorithms, particle swarm optimization algorithms and other types of optimization algorithms [14]. In SOCO, the differential evolution algorithm (DE) [25], the covariance matrix adaptation evolution strategy with increasing population size (G-CMA-ES) [1], and the real-coded CHC algorithm (CHC) [7] are used as the reference algorithms. It should be noted that no ACO-based algorithms are featured in SOCO.

In this paper, we propose an improved $\text{ACO}_{\mathbb{R}}$ algorithm, called $\text{IACO}_{\mathbb{R}}$-LS, that is competitive with the state of the art in continuous optimization. We first present $\text{IACO}_{\mathbb{R}}$, which is an $\text{ACO}_{\mathbb{R}}$ with an extra search diversification mechanism that consists of a growing solution archive. Then, we hybridize $\text{IACO}_{\mathbb{R}}$ with a local search procedure in order to enhance its search intensification abilities. We experiment with three local search procedures: Powell's conjugate directions set [20], Powell's BOBYQA [21], and Lin-Yu Tseng's Mtsls1 [28]. An automatic parameter tuning procedure, Iterated F-race [2,4], is used for the configuration of the investigated algorithms. The best algorithm found after tuning, $\text{IACO}_{\mathbb{R}}$-Mtsls1, obtains results that are as good as the best of the 16 algorithms featured in SOCO. To assess the quality of $\text{IACO}_{\mathbb{R}}$-Mtsls1 and the best SOCO algorithms on problems not seen during their design phase, we compare their performance using a combined benchmark functions suite from the SOCO and the Special Session on Continuous Optimization of the IEEE 2005 Congress on Evolutionary Computation (CEC 2005). The results show that $\text{IACO}_{\mathbb{R}}$-Mtsls1 can be considered to be a state-of-the-art continuous optimization algorithm.

2 The $\text{ACO}_{\mathbb{R}}$ Algorithm

The $\text{ACO}_{\mathbb{R}}$ algorithm stores a set of solutions, called solution archive, which represents the algorithm's "pheromone model." The solution archive is used to create a probability distribution of promising solutions over the search space. Solutions are generated on a coordinate-per-coordinate basis using mixtures of weighted Gaussian functions. Thus, with $\text{ACO}_{\mathbb{R}}$, the exploration of disjoint areas of promising solutions over the search space is possible. An outline of $\text{ACO}_{\mathbb{R}}$ is given in Algorithm 1.

The solution archive consists of k solutions. Initially, it is filled with randomly generated solutions. The algorithm iteratively refines the solution archive by first generating m new solutions and then keeping only the best k solutions of the $k + m$ solutions that are available. The k solutions in the archive are always sorted based on their quality.

The core of the solution construction procedure is the estimation of multimodal one-dimensional probability density functions (PDF). The mechanism to do that in $\text{ACO}_{\mathbb{R}}$ is based on a Gaussian kernel, which is defined as a weighted sum of several Gaussian functions g_j^i, where j is a solution index and i is a

Algorithm 1 Outline of $\text{ACO}_{\mathbb{R}}$

```
// Initialize solutions archive
Initialize solution archive (T) of size k
Evaluate all solutions in T
while Termination criterion not satisfied do
  // Generate m new solutions
  for l = 1 to m do
    // Construct solution l
    for j = 1 to D do
      s_l^j = GenSol(s_j^i, σ_j^i)
    end for
    Evaluate solution S_l
  end for
  // Update solution archive
  T_new = First_k ⇐ Rank(S_1 ··· S_{k+m})
end while
```

coordinate index, as follows:

$$G^i(x) = \sum_{j=1}^{k} \omega_j g_j^i(x) = \sum_{j=1}^{k} \omega_j \frac{1}{\sigma_j^i \sqrt{2\pi}} e^{-\frac{(x-\mu_j^i)^2}{2\sigma_j^{i\,2}}}, \tag{1}$$

where $j \in \{1, ..., k\}$, $i \in \{1, ..., D\}$ with D being the problem dimensionality. ω_j is a weight associated with the rank of solution j in the archive, $rank(j)$. The weight is calculated using a Gaussian function $g(\mu, \sigma) = g(1, qk)$:

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-(rank(j)-1)^2}{2q^2k^2}}, \tag{2}$$

where q is a parameter of the algorithm.

During the solution generation process, each coordinate is treated independently. First, an archive solution is chosen with a probability proportional to its weight. Then, the algorithm samples around the selected solution component s_j^i. This is done using a Gaussian PDF with $\mu_j^i = s_j^i$, and σ_j^i equal to

$$\sigma_j^i \leftarrow \xi \sum_{r=1}^{k} \frac{|s_r^i - s_j^i|}{k-1}, \tag{3}$$

which is the average distance between the i-th variable of the solution s_j and the i-th variable of the other solutions in the archive, multiplied by a parameter ξ. The solution generation process is repeated m times for each dimension $i = 1, ..., D$.

3 The $\text{IACO}_{\mathbb{R}}$-LS Algorithm

The $\text{IACO}_{\mathbb{R}}$-LS algorithm is a hybrid of the Incremental $\text{ACO}_{\mathbb{R}}$ algorithm ($\text{IACO}_{\mathbb{R}}$) and a local search procedure. $\text{IACO}_{\mathbb{R}}$ provides the exploration needed to locate promising solutions, while the local search procedure enables a fast convergence to good solutions. Section 3.1 presents the $\text{IACO}_{\mathbb{R}}$ algorithm and Section 3.2 describes in detail the $\text{IACO}_{\mathbb{R}}$-LS algorithm.

3.1 The IACO$_{\mathbb{R}}$ algorithm

IACO$_{\mathbb{R}}$ is an ACO$_{\mathbb{R}}$ algorithm with a solution archive whose size increases as the optimization process evolves. This modification is based on the incremental social learning framework [17,18]. The effect of this strategy depends on the rate at which the archive increases its size. Fast rates encourage search diversification while slow ones encourage intensification [18]. Instead of a parameter that determines the archive size, a parameter *growth* controls the rate at which the archive grows. In IACO$_{\mathbb{R}}$ the optimization process begins with a small archive and a new solution is added to it every *growth* iterations until a maximum archive size is reached. Each time a new solution is added, it is initialized using information from other solutions in the archive. First, a new solution $\boldsymbol{S}_{\text{new}}$ is generated completely at random. Then, it is moved toward the best solution in the archive $\boldsymbol{S}_{\text{best}}$ using

$$\boldsymbol{S}'_{\text{new}} = \boldsymbol{S}_{\text{new}} + \text{rand}(0,1)(\boldsymbol{S}_{\text{best}} - \boldsymbol{S}_{\text{new}})\,, \tag{4}$$

where rand$(0,1)$ is a random number in the range $[0,1)$.

In addition to the growing solution archive, IACO$_{\mathbb{R}}$ features a different mechanism for selecting the solution that guides the generation of new solutions. The new procedure depends on a parameter $p \in [0,1]$, which controls the probability of using only the best solution in the archive. With a probability $1-p$, all the solutions in the archive are used to generate new solutions. Once a guiding solution is selected, and a new one is generated (in exactly the same way as in ACO$_{\mathbb{R}}$), they are compared. If the newly generated solution is better than the guiding solution, it replaces it in the archive. This replacement strategy is different from the one used in ACO$_{\mathbb{R}}$ in which all the solutions in the archive and all the newly generated ones compete.

For fighting stagnation, we include an algorithm-level diversification mechanism that consists in restarting the algorithm and initializing the new initial archive with the best-so-far solution. The restart criterion is the number of consecutive iterations $MaxStagIter$ with a relative solution improvement lower than a certain threshold.

3.2 Hybrid IACO$_{\mathbb{R}}$ with Local Search

IACO$_{\mathbb{R}}$-LS is a hybrid algorithm composed of IACO$_{\mathbb{R}}$ and a local search procedure. IACO$_{\mathbb{R}}$-LS works as follows: First, the solution archive is initialized. Then, the local search procedure is called using the best solution in the archive as initial point. After that, IACO$_{\mathbb{R}}$ is executed. The outline of IACO$_{\mathbb{R}}$-LS algorithm is shown in Algorithm 2.

In our experiments, we considered Powell's conjugate directions set [20], Powell's BOBYQA [21] and Lin-Yu Tseng's Mtsls1 [28] methods as local search procedures. We used the NLopt library implementation of the first two methods and implemented Mtsls1 following the pseudocode found in [28]. The local search methods terminate after a maximum number of iterations, MaxITER, have been reached, or when the tolerance FTOL, that is the relative change between solutions found in two consecutive iterations, falls below a certain threshold. Like [16], we use an adaptive step size for the local search procedures. This is achieved as follows: a solution in the archive, different from the best solution, is chosen at random. The maximum norm ($||\cdot||_\infty$) of the vector that separates

Algorithm 2 Outline of IACO$_{\mathbb{R}}$-LS

Parameters: Parameters of IACO$_{\mathbb{R}}$: p, ξ. Parameters of local search: *FTOL*, *MaxITER*, *MaxFailures*. Parameters of restart mechanism: *MaxStagIter*. Parameters of growing archive: *InitArchiveSize*, *growth*.
// Initialize solutions archive
Initialize solution archive (T) of size $k \leftarrow$ *InitArchiveSize*
Evaluate all solutions in T
while Termination criterion not satisfied **do**
  // Local search
  if *FailedAttempts* = *MaxFailures* **then**
    Invoke local search on $\boldsymbol{S}_{\text{rand}} \in T$
  else
    Invoke local search on $\boldsymbol{S}_{\text{best}} \in T$
  end if

  // Generate new solutions
  if rand(0,1)¡p **then**
    // Select best solution
    for $j = 1$ to D **do**
      $s^j_{\text{new}} = \text{GenSol}(s^{\text{best}}_j, \sigma^{\text{best}}_j)$
    end for
    Evaluate solution $\boldsymbol{S}_{\text{new}}$
    if $\boldsymbol{S}_{\text{new}}$ is better than $\boldsymbol{S}_{\text{best}}$ **then**
      Substitute $\boldsymbol{S}_{\text{best}}$ for $\boldsymbol{S}_{\text{new}}$ in T
    end if
  else
    // Select all k solutions
    for $l = 1$ to k **do**
      for $j = 1$ to D **do**
        $s^j_{\text{new}} = \text{GenSol}(s^l_j, \sigma^l_j)$
      end for
      Evaluate solution $\boldsymbol{S}_{\text{new}}$
      if $\boldsymbol{S}_{\text{new}}$ is better than $\boldsymbol{S}_l$ **then**
        Substitute $\boldsymbol{S}_l$ for $\boldsymbol{S}_{\text{new}}$ in T
      end if
    end for
  end if

  // Archive growth
  if Solution addition criterion is met **then**
    $\boldsymbol{S}'_{\text{new}} = \boldsymbol{S}_{\text{new}} + rand(0,1)(\boldsymbol{S}_{\text{best}} - \boldsymbol{S}_{\text{new}})$
    $T \leftarrow T \cup \boldsymbol{S}'_{\text{new}}$
    $k \leftarrow k + 1$
  end if

  // Restart Mechanism
  if *noImprovement* = *MaxStagIter* **then**
    Re-initialize T but keeping $\boldsymbol{S}_{\text{best}}$
  end if
end while

this random solution from the best solution is used as the local search step size. Hence, step sizes tend to decrease over time due to the convergence tendency of the solutions in the archive. This phenomenon in turn makes the search focus around the best-so-far solution.

For fighting stagnation at the level of the local search, we call the local search procedure from different solutions from time to time. A parameter, *MaxFailures*, determines the maximum number of repeated calls to the local search method from the same initial solution that does not result in a solution improvement. We maintain a failures counter for each solution in the archive. The local search procedure cannot be called again from a solution if its associated counter reaches the value of *MaxFailures* In that case, the local search is called from a random solution as long as this random solution's failures counter is less than *MaxFailures*.

Finally, we use a simple mechanism to enforce boundary constraints in IACO$_{\mathbb{R}}$-LS. We use the following penalty function in Powell's conjugate directions method as well as in Mtsls1:

$$P(\boldsymbol{x}) = fes \cdot \sum_{i=1}^{D} Bound(x_i)\,, \tag{5}$$

where $Bound(x_i)$ is defined as

$$Bound(x_i) = \begin{cases} 0, & \text{if } x_{\min} \leq x_i \leq x_{\max} \\ (x_{\min} - x_i)^2, & \text{if } x_i < x_{\min} \\ (x_{\max} - x_i)^2, & \text{if } x_i > x_{\max} \end{cases} \tag{6}$$

where x_{min} and x_{max} are the minimum and maximum limits of the search range, respectively, and *fes* is the number of function evaluations that have been used so far. BOBYQA has its own mechanism for dealing with bound constraints.

4 Experimental Study

Our study is carried out in two stages. In the first stage, we evaluate the performance of ACO$_{\mathbb{R}}$, IACO$_{\mathbb{R}}$-BOBYQA, IACO$_{\mathbb{R}}$-Powell and IACO$_{\mathbb{R}}$-Mtsls1 by comparing their performance with that of the 16 algorithms featured in SOCO. For this purpose, we use the same 19 benchmark functions suite (functions labeled as $f_{soco}*$). In the second stage, we include 21[1] of the benchmark functions proposed for the special session on continuous optimization organized for the IEEE 2005 Congress on Evolutionary Computation (CEC 2005) [26] (functions labeled as $f_{cec}*$).

In the first stage of the study, we used the 50- and 100-dimensional versions of the 19 f_{soco} functions. Functions f_{soco1}–f_{soco6} were originally proposed for the special session on large scale global optimization organized for the IEEE 2008 Congress on Evolutionary Computation (CEC 2008) [27]. Functions f_{soco7}-f_{soco11} were proposed at the ISDA 2009 Conference. Functions f_{soco12}-f_{soco19} are hybrid functions that combine two functions belonging to f_{soco1}–f_{soco11}. The detailed description of these functions is available in [9, 14]. In the second

[1] From the original 25 functions, we decided to omit f_{cec1}, f_{cec2}, f_{cec6}, and f_{cec9} because they are the same as f_{soco1}, f_{soco3}, f_{soco4}, f_{soco8}.

Table 1: Benchmark functions

ID	Name/Description	Uni/Multi modal	Separable	Rotated
f_{soco1}	Shift.Sphere	U	Y	N
f_{soco2}	Shift.Schwefel 2.21	U	N	N
f_{soco3}	Shift.Rosenbrock	M	N	N
f_{soco4}	Shift.Rastrigin	M	Y	N
f_{soco5}	Shift.Griewank	M	N	N
f_{soco6}	Shift.Ackley	M	Y	N
f_{soco7}	Shift.Schwefel 2.22	U	Y	N
f_{soco8}	Shift.Schwefel 1.2	U	N	N
f_{soco9}	Shift.Extended f_{10}	U	N	N
f_{soco10}	Shift.Bohachevsky	U	N	N
f_{soco11}	Shift.Schaffer	U	N	N
f_{soco12}	$f_{soco9} \oplus_{0.25} f_{soco1}$	M	N	N
f_{soco13}	$f_{soco9} \oplus_{0.25} f_{soco3}$	M	N	N
f_{soco14}	$f_{soco9} \oplus_{0.25} f_{soco4}$	M	N	N
f_{soco15}	$f_{soco10} \oplus_{0.25} f_{soco7}$	M	N	N
f_{soco16}	$f_{soco9} \oplus_{0.5} f_{soco1}$	M	N	N
f_{soco17}	$f_{soco9} \oplus_{0.75} f_{soco3}$	M	N	N
f_{soco18}	$f_{soco9} \oplus_{0.75} f_{soco4}$	M	N	N
f_{soco19}	$f_{soco10} \oplus_{0.75} f_{soco7}$	M	N	N
f_{cec3}	Shift.Ro..Elliptic	U	N	Y
f_{cec4}	Shift.Schwefel 1.2 Noise	U	N	N
f_{cec5}	Schwefel 2.6 Opt on Bound	U	N	N
f_{cec7}	Shift.Ro.Griewank No Bound	M	N	Y
f_{cec8}	Shift.Ro.Ackley Opt on Bound	M	N	Y
f_{cec10}	Shift.Ro.Rastrigin	M	N	Y
f_{cec11}	Shift.Ro.Weierstrass	M	N	Y
f_{cec12}	Schwefel 2.13	M	N	N
f_{cec13}	Griewank plus Rosenbrock	M	N	N
f_{cec14}	Shift.Ro.Exp.Scaffer	M	N	Y
f_{cec15}	Hybrid Composition	M	N	N
f_{cec16}	Ro. Hybrid Composition	M	N	Y
f_{cec17}	Ro. Hybrid Composition	M	N	Y
f_{cec18}	Ro. Hybrid Composition	M	N	Y
f_{cec19}	Ro. Hybrid Composition	M	N	Y
f_{cec20}	Ro. Hybrid Composition	M	N	Y
f_{cec21}	Ro. Hybrid Composition	M	N	Y
f_{cec22}	Ro. Hybrid Composition	M	N	Y
f_{cec23}	Ro. Hybrid Composition	M	N	Y
f_{cec24}	Ro. Hybrid Composition	M	N	Y
f_{cec25}	Ro. Hybrid Composition	M	N	Y

stage of our study, the 19 f_{soco} and 21 f_{cec} functions on 50 dimensions were considered together. Some properties of the benchmark functions are listed in Table 1. The detailed description is available in [9, 26].

We used the same termination conditions defined for SOCO and CEC 2005, that is, the maximum number of function evaluations was $5000 \times D$ for the SOCO functions, and $10000 \times D$ for the CEC 2005 functions. All the investigated algorithms were run 25 times on each function. For a solution $\boldsymbol{x}$, the error is measured as $f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$, where $\boldsymbol{x}^*$ is the optimum of the function. Error values lower than 10^{-14} are approximated to 0. Our analysis is based on either the whole solution quality distribution, or on the median and average errors.

4.1 Parameter settings

We used Iterated F-race [2, 4] as the automatic parameter tuning procedure. The 10-dimensional versions of the 19 f_{soco} were used as training instances. A maximum of 50000 algorithm runs were used as tuning budget for $\text{ACO}_{\mathbb{R}}$, $\text{IACO}_{\mathbb{R}}$-BOBYQA, $\text{IACO}_{\mathbb{R}}$-Powell and $\text{IACO}_{\mathbb{R}}$-Mtsls1. The best found set of parameters for each algorithm is given in Table 2.

Table 2: Best parameter settings found through iterated F-Race for $ACO_{\mathbb{R}}$, $IACO_{\mathbb{R}}$-BOBYQA, $IACO_{\mathbb{R}}$-Powell and $IACO_{\mathbb{R}}$-Mtsls1

$ACO_{\mathbb{R}}$	q	ξ	*mants*	k				
	0.04544	0.8259	10	85				
$IACO_{\mathbb{R}}$-BOBYQA	p	ξ	initarchsize	Growth	FTol	MaxITER	MaxFailures	MaxStagIter
	0.6979	0.8643	4	1	-3.13	240	5	20
$IACO_{\mathbb{R}}$-Powell	p	ξ	initarchsize	Growth	FTol	MaxITER	MaxFailures	MaxStagIter
	0.3586	0.9040	1	7	-1	20	6	8
$IACO_{\mathbb{R}}$-Mtsls1	p	ξ	initarchsize	Growth	MaxITER	MaxFailures	MaxStagIter	
	0.6475	0.7310	14	1	85	4	13	

4.2 Experimental results and comparison

Figure 1 shows the distribution of median and average errors across the 19 f_{soco} benchmark functions obtained with $ACO_{\mathbb{R}}$, $IACO_{\mathbb{R}}$-BOBYQA, $IACO_{\mathbb{R}}$-Powell, $IACO_{\mathbb{R}}$-Mtsls1 and the 16 algorithms featured in SOCO.[2] We marked with a + symbol those cases in which there is a statistically significant difference at the 5% level with respect to $IACO_{\mathbb{R}}$-Mtsls1 (in favor of $IACO_{\mathbb{R}}$-Mtsls1). Also at the top of each plot, a count of the number of optima found by each algorithm (or an ojective function value lower than 10^{-14}) is given.

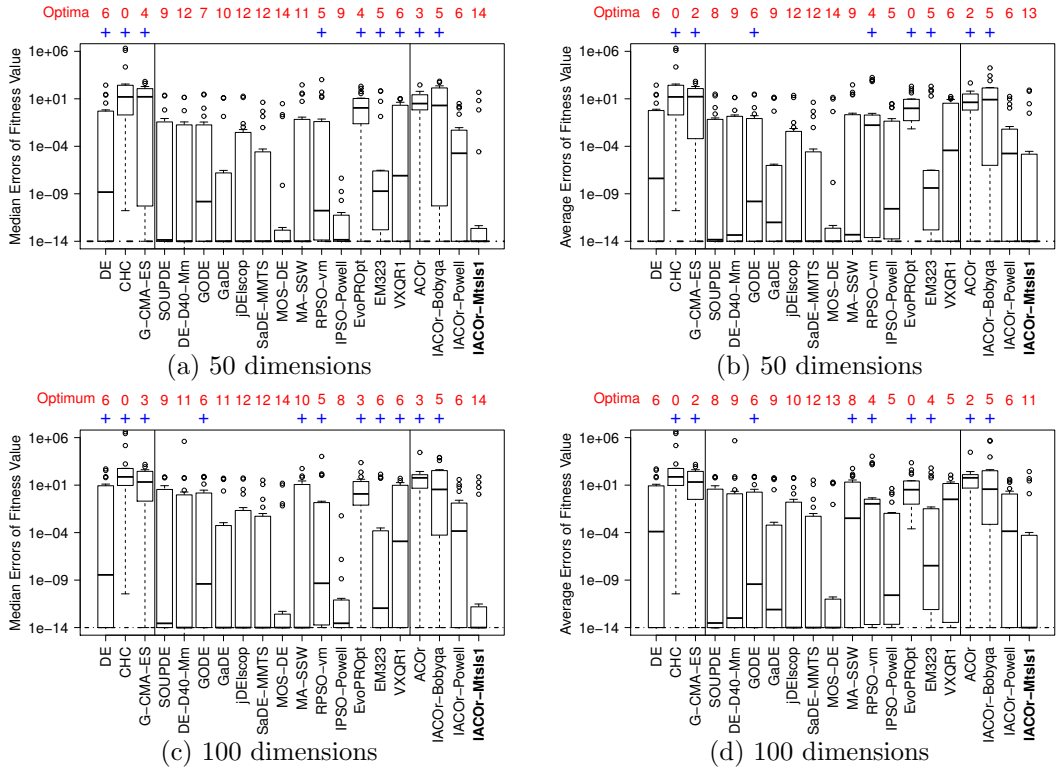

Figure 1: The box-plots show the distribution of the median and average errors obtained on the 19 SOCO benchmark functions. The results obtained with the three reference algorithms in SOCO are shown on the left part of each plot. The results of 13 algorithms published in SOCO are shown in the middle part of each plot. The results obtained with $ACO_{\mathbb{R}}$, $IACO_{\mathbb{R}}$-BOBYQA, $IACO_{\mathbb{R}}$-Powell, and $IACO_{\mathbb{R}}$-Mtsls1 are shown on the right part of each plot. The line at the bottom of each plot represents the 0-threshold (10^{-14}). A + symbol on top of a box-plot denotes a statistically significant difference at the 5% level detected with a Wilcoxon test between the results obtained with the indicated algorithm and those obtained with $IACO_{\mathbb{R}}$-Mtsls1. The numbers on top of a box-plot denotes the number of optima found by the corresponding algorithm.

In all cases, $IACO_{\mathbb{R}}$-Mtsls1 significantly outperforms $ACO_{\mathbb{R}}$, and is in general more effective than $IACO_{\mathbb{R}}$-BOBYQA, and $IACO_{\mathbb{R}}$-Powell. $IACO_{\mathbb{R}}$-Mtsls1

[2]For information about these 16 algorithms please go to `http://sci2s.ugr.es/eamhco/CFP.php`

Table 3: The one-sided p-values of Wilcoxon matched-pairs signed-rank test of IACO$_{\mathbb{R}}$-Mtsls1 with other algorithms. The p-value (¡0.05) is highlighted in **boldface**.

Algs	Median Errors		Average Errors	
	D50	D100	D50	D100
DE	**4.7E−02**	**7.2E−03**	1.5E−01	6.2E−02
CHC	**1.9E−06**	**1.9E−06**	**1.9E−06**	**1.9E−06**
G−CMA−ES	**1.4E−03**	**1.2E−03**	**9.9E−04**	**1.8E−03**
SOUPDE	1.8E−01	9.3E−02	2.8E−01	2.9E−01
DE−D40−Mm	3.8E−01	2.3E−01	3.8E−01	1.5E−01
GODE	8.5E−02	**7.2E−03**	2.4E−01	2.4E−01
GaDE	4.1E−01	3.6E−01	6.2E−01	6.2E−01
jDElscop	4.0E−01	3.4E−01	6.4E−01	4.8E−01
SaDE−MMTS	4.5E−01	5.5E−01	6.8E−01	5.8E−01
MOS−DE	9.9E−01	9.9E−01	9.9E−01	9.9E−01
MA−SSW	1.7E−01	**1.7E−02**	2.5E−01	**2.5E−02**
RPSO−vm	**6.0E−03**	**5.5E−04**	**2.5E−03**	**8.1E−04**
Tuned−IPSOLS	7.2E−01	7.0E−01	3.0E−01	5.1E−01
EVoPROpt	**3.6E−04**	**2.4E−04**	**1.9E−06**	**1.9E−06**
EM323	**5.5E−04**	**8.3E−04**	**3.0E−02**	**1.7E−02**
VXQR1	**1.6E−02**	**1.9E−02**	1.4E−01	5.3E−02
ACO$_{\mathbb{R}}$	**1.7E−03**	**2.4E−04**	**9.0E−03**	**1.7E−03**
IACO$_{\mathbb{R}}$−Bobyqa	**5.5E−04**	**5.5E−04**	**5.5E−04**	**4.5E−04**
IACO$_{\mathbb{R}}$−Powell	8.1E−02	2.2E−01	4.7E−01	2.9E−01

is also competitive with the best algorithms in SOCO. If we consider medians only, IACO$_{\mathbb{R}}$-Mtsls1 significantly outperforms G-CMA-ES, CHC, DE, EVoPROpt, VXQR1, EM323, and RPSO-vm in both 50 and 100 dimensions. In 100 dimensions, IACO$_{\mathbb{R}}$-Mtsls1 also significantly outperforms MA-SSW and GODE. Moreover, the median error of IACO$_{\mathbb{R}}$-Mtsls1 is below the 0-threshold 14 times out of the 19 possible of the SOCO benchmark functions suite. Only MOS-DE matches such a performance.

If one considers mean values, the performance of IACO$_{\mathbb{R}}$-Mtsls1 degrades slightly. This is an indication that IACO$_{\mathbb{R}}$-Mtsls1 still stagnates with some low probability. However, IACO$_{\mathbb{R}}$-Mtsls1 still outperforms G-CMA-ES, CHC, GODE, EVoPROpt, RPSO-vm, and EM323. Even though IACO$_{\mathbb{R}}$-Mtsls1 does not significantly outperform DE and other algorithms, its performance is very competitive. The mean error of IACO$_{\mathbb{R}}$-Mtsls1 is below the 0-threshold 13 and 11 times in problems of 50 and 100 dimensions, respectively.

The one-sided p-values of Wilcoxon matched-pairs signed-rank test of IACO$_{\mathbb{R}}$-Mtsls1 with other algorithms at 5% level is listed in Table 3.

We note that although G-CMA-ES has difficulties in dealing with multimodal or unimodal shifted separable functions, such as f_{soco4} , f_{soco6} and f_{soco7}, G-CMA-ES showed impressive results on function f_{soco8}, which is a hyperellipsoid rotated in all directions. None of other investigated algorithms can find the optimum of this function except G-CMA-ES. This result is interesting considering that G-CMA-ES showed an impressive performance in the CEC 2005 special session on continuous optimization. This fact suggests that releasing details about the problems that will be used to compare algorithms induces an undesired "overfitting" effect. In other words, authors may use the released

problems to design algorithms that perform well on them but that may perform poorly on another unknown set of problems. This motivated us to carry out the second stage of our study, which is presented next.

The second stage of our study consists in carrying out a more comprehensive comparison that includes G-CMA-ES and some of the best algorithms in SOCO. For this comparison, we use 40 benchmark functions as discussed above. From SOCO, we include in our study IPSO-Powell given its good performance as shown in Figure 1. To discard the possibility that the local search procedure is the main responsible for the obtained results, we also use Mtsls1 with IPSO, thus generating IPSO-Mtsls1. In this second stage, IPSO-Powell and IPSO-Mtsls1 were tuned as described in Section 4.1.

Table 4 shows the median and average errors obtained by the compared algorithm on each of the 40 benchmark functions. Two facts can be noticed from these results. First, Mtsls1 seems to be indeed responsible for most of the good performance of the algorithms that use it as a local search procedure. Regarding median results, the SOCO functions for which IPSO-Mtsls1 finds the optimum, $\text{IACO}_{\mathbb{R}}$-Mtsls1 does it as well. However, $\text{IACO}_{\mathbb{R}}$-Mtsls1 seems to be more robust given the fact that it finds more optima than IPSO-Mtsls1 if functions from the CEC 2005 special session or mean values are considered. Second, G-CMA-ES finds more best results on the CEC 2005 functions than on the SOCO functions. This result adds evidence in favor of the overfitting effect mentioned earlier. Overall, however, $\text{IACO}_{\mathbb{R}}$-Mtsls1 finds more best results than any of the compared algorithms.

Figure 2 shows correlation plots that illustrate the relative performance between $\text{IACO}_{\mathbb{R}}$-Mtsls1 and the other three algorithms. On the x-axis, the coordinates are the results obtained with $\text{IACO}_{\mathbb{R}}$-Mtsls1; on the y-axis, the coordinates are the results obtained with another algorithm for each of the 40 functions. Thus, points that appear on the left part of the correlation plot correspond to functions for which $\text{IACO}_{\mathbb{R}}$-Mtsls1 has better results than the other algorithm.

Table 5 shows a detailed comparison presented in form of (win, draw, lose) according to different properties of the 40 functions used. The one-sided p-values of Wilcoxon matched-pairs signed-rank test of $\text{IACO}_{\mathbb{R}}$-Mtsls1 with other algorithms across 40 functions are also presented. In general, $\text{IACO}_{\mathbb{R}}$-Mtsls1 outperforms more often all the other compared algorithms. $\text{IACO}_{\mathbb{R}}$-Mtsls1 wins more often against G-CMA-ES; however, G-CMA-ES performs clearly better than $\text{IACO}_{\mathbb{R}}$-Mtsls1 on rotated functions, which can be explained by the covariance matrix adaptation mechanism [8].

5 Conclusions and Future Work

In this paper, we have introduced $\text{IACO}_{\mathbb{R}}$-LS, an $\text{ACO}_{\mathbb{R}}$ algorithm with growing solution archive hybridized with a local search procedure. Three different local search procedures, Powell's conjugate directions set, Powell's BOBYQA, and Mtsls1, were tested with $\text{IACO}_{\mathbb{R}}$-LS. Through automatic tuning across 19 functions, $\text{IACO}_{\mathbb{R}}$-Mtsls1 proved to be superior than the other two variants.

The results of a comprehensive experimental comparison with 16 algorithm featured in a recent special issue of the journal Soft Computing show that $\text{IACO}_{\mathbb{R}}$-Mtsls1 greatly outperforms the original $\text{ACO}_{\mathbb{R}}$ and that $\text{IACO}_{\mathbb{R}}$-Mtsls1 is competitive with the state of the art. A second comparison that included

Table 4: The median and average errors of objective function values obtained with G-CMA-ES, IPSO-Powell, IPSO-Mtsls1, and IACO$_{\mathbb{R}}$-Mtsls1 on 40 funtions with dimensions D = 50. The lowest values were highlighted in **boldface**. The values below 10^{-14} are approximated to 0. The results of f_{cec1}, f_{cec2}, f_{cec6}, f_{cec9} are not presented to void repeated test on the similar functions such as f_{soco1}, f_{soco3}, f_{soco4}, f_{soco8}. On the bottom of the table, the number of finding the lowest errors among the investigated algorithms are counted.

	Median errors					Mean errors			
F_i	G-CMA-ES	IPSO-Powell	IPSO-Mtsls1	IACO$_{\mathbb{R}}$-Mtsls1	F_i	G-CMA-ES	IPSO-Powell	IPSO-Mtsls1	IACO$_{\mathbb{R}}$-Mtsls1
f_{soco1}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco1}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{soco2}	2.64E−11	**1.42E−14**	4.12E−13	4.41E−13	f_{soco2}	2.75E−11	**2.56E−14**	4.80E−13	5.50E−13
f_{soco3}	**0.00E+00**	**0.00E+00**	6.38E+00	4.83E+01	f_{soco3}	7.97E−01	**0.00E+00**	7.29E+01	8.17E+01
f_{soco4}	1.08E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco4}	1.05E+02	**0.00E+00**	1.31E+00	**0.00E+00**
f_{soco5}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco5}	2.96E−04	6.72E−03	5.92E−04	**0.00E+00**
f_{soco6}	2.11E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco6}	2.09E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{soco7}	7.67E−11	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco7}	1.01E−10	4.98E−12	**0.00E+00**	**0.00E+00**
f_{soco8}	**0.00E+00**	1.75E−09	2.80E−10	2.66E−05	f_{soco8}	**0.00E+00**	4.78E−09	4.29E−10	2.94E−05
f_{soco9}	1.61E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco9}	1.66E+01	4.95E−06	**0.00E+00**	**0.00E+00**
f_{soco10}	6.71E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco10}	6.81E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{soco11}	2.83E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco11}	3.01E+01	8.19E−02	7.74E−02	**0.00E+00**
f_{soco12}	1.87E+02	1.02E−12	**0.00E+00**	**0.00E+00**	f_{soco12}	1.88E+02	1.17E−11	7.27E−03	**0.00E+00**
f_{soco13}	1.97E+02	**2.00E−10**	5.39E−01	6.79E−01	f_{soco13}	1.97E+02	**2.65E−10**	2.75E+00	3.03E+00
f_{soco14}	1.05E+02	1.77E−12	**0.00E+00**	**0.00E+00**	f_{soco14}	1.09E+02	1.18E+00	5.26E−01	**3.04E−01**
f_{soco15}	8.12E−04	1.07E−11	**0.00E+00**	**0.00E+00**	f_{soco15}	9.79E−04	2.62E−11	**0.00E+00**	**0.00E+00**
f_{soco16}	4.22E+02	3.08E−12	**0.00E+00**	**0.00E+00**	f_{soco16}	4.27E+02	2.80E+00	2.46E+00	**0.00E+00**
f_{soco17}	6.71E+02	**4.35E−08**	1.47E+01	6.50E+00	f_{soco17}	6.89E+02	**3.10E+00**	7.27E+01	6.19E+01
f_{soco18}	1.27E+02	8.06E−12	**0.00E+00**	**0.00E+00**	f_{soco18}	1.31E+02	1.24E+00	1.68E+00	**0.00E+00**
f_{soco19}	4.03E+00	1.83E−12	**0.00E+00**	**0.00E+00**	f_{soco19}	4.76E+00	1.19E−11	**0.00E+00**	**0.00E+00**
f_{cec3}	**0.00E+00**	8.72E+03	1.59E+04	8.40E+05	f_{cec3}	**0.00E+00**	1.24E+04	1.62E+04	9.66E+05
f_{cec4}	4.27E+05	2.45E+02	3.88E+03	**5.93E+01**	f_{cec4}	4.68E+05	2.90E+02	4.13E+03	**7.32E+01**
f_{cec5}	5.70E−01	4.87E−07	**7.28E−11**	9.44E+00	f_{cec5}	2.85E+00	4.92E−06	**2.32E−10**	9.98E+00
f_{cec7}	3.85E−14	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{cec7}	5.32E−14	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{cec8}	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	f_{cec8}	2.01E+01	**2.00E+01**	**2.00E+01**	**2.00E+01**
f_{cec10}	**9.97E−01**	8.96E+02	8.92E+02	2.69E+02	f_{cec10}	**1.72E+00**	9.13E+02	8.76E+02	2.75E+02
f_{cec11}	**1.21E+00**	6.90E+01	6.64E+01	5.97E+01	f_{cec11}	**1.17E+01**	6.82E+01	6.63E+01	5.90E+01
f_{cec12}	**2.36E+03**	5.19E+04	3.68E+04	1.37E+04	f_{cec12}	2.27E+05	5.68E+04	5.86E+04	**1.98E+04**
f_{cec13}	4.71E+00	3.02E+00	3.24E+00	**2.14E+00**	f_{cec13}	4.59E+00	3.18E+00	3.32E+00	**2.13E+00**
f_{cec14}	**2.30E+01**	2.35E+01	2.36E+01	2.33E+01	f_{cec14}	**2.29E+01**	2.34E+01	2.35E+01	2.31E+01
f_{cec15}	2.00E+02	2.00E+02	2.00E+02	**0.00E+00**	f_{cec15}	2.04E+02	1.82E+02	2.06E+02	**9.20E+01**
f_{cec16}	**2.15E+01**	4.97E+02	4.10E+02	3.00E+02	f_{cec16}	**3.09E+01**	5.22E+02	4.80E+02	3.06E+02
f_{cec17}	**1.61E+02**	4.54E+02	4.11E+02	4.37E+02	f_{cec17}	**2.34E+02**	4.46E+02	4.17E+02	4.43E+02
f_{cec18}	**9.13E+02**	1.22E+03	1.21E+03	9.84E+02	f_{cec18}	**9.13E+02**	1.18E+03	1.19E+03	9.99E+02
f_{cec19}	**9.12E+02**	1.23E+03	1.19E+03	9.93E+02	f_{cec19}	**9.12E+02**	1.22E+03	1.18E+03	1.01E+03
f_{cec20}	**9.12E+02**	1.22E+03	1.19E+03	9.93E+02	f_{cec20}	**9.12E+02**	1.20E+03	1.18E+03	9.89E+02
f_{cec21}	1.00E+03	1.19E+03	1.03E+03	**5.00E+02**	f_{cec21}	1.00E+03	9.86E+02	8.59E+02	**5.53E+02**
f_{cec22}	**8.03E+02**	1.43E+03	1.45E+03	1.13E+03	f_{cec22}	**8.05E+02**	1.45E+03	1.47E+03	1.14E+03
f_{cec23}	1.01E+03	**5.39E+02**	**5.39E+02**	**5.39E+02**	f_{cec23}	1.01E+03	7.66E+02	6.13E+02	**5.67E+02**
f_{cec24}	**9.86E+02**	1.31E+03	1.30E+03	1.11E+03	f_{cec24}	**9.55E+02**	1.29E+03	1.30E+03	1.10E+03
f_{cec25}	**2.15E+02**	1.50E+03	1.59E+03	9.38E+02	f_{cec25}	**2.15E+02**	1.18E+03	1.50E+03	8.89E+02
No.of.best	18	15	18	21	No.of.best	14	10	10	22

21 extra functions from the special session on continuous optimization of the IEEE 2005 Congress on Evolutionary Computation (CEC 2005), showed that IACO$_{\mathbb{R}}$-Mtsls1 obtains higher quality solutions than G-CMA-ES on functions with different properties except on those that are rotated. From among the 20 compared algorithms, only G-CMA-ES obtained good results on rotated

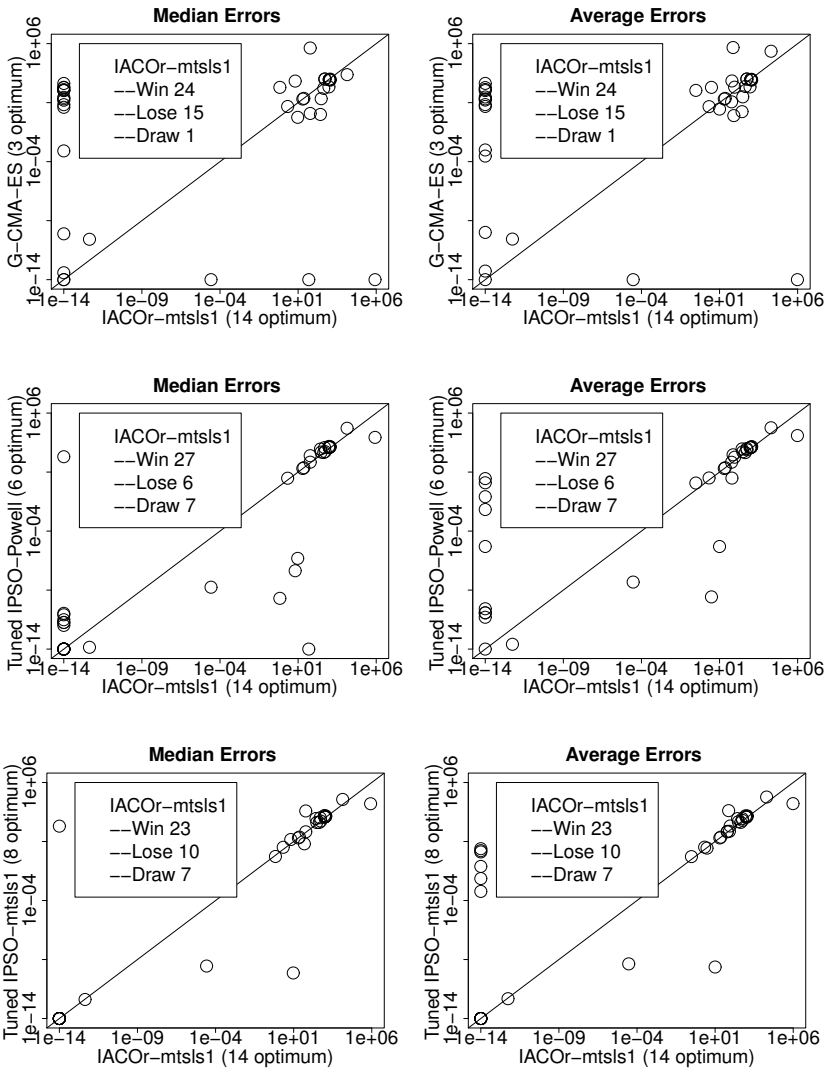

Figure 2: The correlation plot between IACO$_{\mathbb{R}}$-Mtsls1 and G-CMA-ES, IPSO-Powell and IPSO-Mtsls1 over 40 functions. Each point represents a function. The points on the left part of correlation plot illustrate that on those represented functions, IACO$_{\mathbb{R}}$-Mtsls1 obtains better results than the other algorithm.

functions. Although G-CMA-ES obtain good results on most of the rotated benchmark functions of the CEC 2005 special session, it showed a worse performance on most of the Soft Computing non-rotated benchmark functions. On the contrary, IACO$_{\mathbb{R}}$-Mtsls1 has a good performance on most of he Soft Computing benchmark functions, but showed worse performance when compared to G-CMA-ES on most of the rotated benchmark functions of the CEC 2005 special session. This fact suggests that releasing details about the problems that will be used to compare algorithms induces an undesired "overfitting" effect. In other words, authors may use the released problems to design algorithms that

Table 5: The comparison is conducted based on median and average errors of objective value and the results of $\text{IACO}_{\mathbb{R}}$-Mtsls1 are presented in form of (win, draw, lose), respectively. The tested 40 functions were divided into different properties for details. The one-sided p-values of Wilcoxon matched-pairs signed-rank test of $\text{IACO}_{\mathbb{R}}$-Mtsls1 at a 5% level with other algorithms are also presented

Median Errors			
Properties of Functions	$\text{IACO}_{\mathbb{R}}$-Mtsls1 vs G-CMA-ES	$\text{IACO}_{\mathbb{R}}$-Mtsls1 vs IPSO-Powell	$\text{IACO}_{\mathbb{R}}$-Mtsls1 vs IPSO-Mtsls1
Separable	(3, 1, 0)	(0, 4, 0)	(0, 4, 0)
Non-Separable	(18, 2, 16)	(22, 7, 7)	(16, 13, 7)
Non-Separable (Non-Hybrid)	(7, 2, 8)	(6, 6, 5)	(6, 6, 5)
Non-Separable (Hybrid)	(11, 0, 8)	(16, 1, 2)	(10, 7, 2)
Unimodal	(6, 1, 3)	(1, 5, 4)	(1, 5, 4)
Multimodal	(15, 2, 13)	(21, 6, 3)	(15, 12, 3)
Non-rotated	(16, 2, 6)	(10, 8, 6)	(10, 8, 6)
Rotated	(5, 1, 10)	(12, 3, 1)	(12, 3, 1)
SOCO	(15, 2, 2)	(6, 8, 5)	(1, 14, 4)
CEC 2005	(6, 1, 14)	(16, 3, 2)	(15, 3, 3)
In total	(21, 3, 16)	(22, 11, 7)	(16, 17, 7)
p-value	4.16E−01	3.01E−03	6.59E−03
Average Errors			
Properties of Functions	$\text{IACO}_{\mathbb{R}}$-Mtsls1 vs G-CMA-ES	$\text{IACO}_{\mathbb{R}}$-Mtsls1 vs IPSO-Powell	$\text{IACO}_{\mathbb{R}}$-Mtsls1 vs IPSO-Mtsls1
Separable	(3, 1, 0)	(1, 3, 0)	(1, 3, 0)
Non-Separable	(21, 0, 15)	(26, 3, 7)	(23, 6, 7)
Non-Separable (Non-Hybrid)	(10, 0, 7)	(9, 3, 5)	(8, 4, 5)
Non-Separable (Hybrid)	(11, 0, 8)	(17, 0, 2)	(15, 2, 2)
Unimodal	(6, 1, 3)	(4, 2, 4)	(2, 4, 4)
Multimodal	(18, 0, 12)	(23, 4, 3)	(22, 5, 3)
Non-rotated	(20, 1, 3)	(13, 5, 6)	(21, 7, 6)
Rotated	(4, 0, 12)	(14, 1, 1)	(13, 2, 1)
SOCO	(16, 1, 2)	(10, 4, 5)	(8, 7, 4)
CEC 2005	(8, 0, 13)	(17, 2, 2)	(16, 2, 3)
In total	(24, 1, 15)	(27, 6, 7)	(24, 9, 7)
p-value	2.11E−01	9.30E−04	8.30E−04

perform well on them but that may perform poorly on another unknown set of problems.

References

[1] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 1769–1776, Piscataway, NJ, 2005. IEEE Press.

[2] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In *LNCS*

4771. Proceedings of the 4th International Conference on Hybrid Metaheuristics, pages 108–122, Berlin, Germany, 2007. Springer.

[3] G. Bilchev and I. Parmee. The Ant Colony Metaphor for Searching Continuous Design Spaces. In T. Fogarty, editor, *AISB Workshop on Evolutionary Computing*, pages 25–39. Springer-Verlag, 1995.

[4] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. F-Race and iterated F-Race: An overview. *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336, 2010.

[5] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[6] J. Dréo and P. Siarry. Continuous interacting ant colony algorithm based on dense heterarchy. *Future Generation Computer Systems*, 20(5):841–856, 2004.

[7] L. Eshelman and J. Schaffer. Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms*, 2(1993):187–202, 1993.

[8] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 57–64, San Francisco, CA, 1995. Morgan Kaufmann.

[9] F. Herrera, M. Lozano, and D. Molina. Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems, 2010. URL: `http://sci2s.ugr.es/eamhco/updated-functions1-19.pdf`.

[10] X. Hu, J. Zhang, and Y. Li. Orthogonal methods based ant colony search for solving continuous optimization problems. *Journal of Computer Science and Technology*, 23(1):2–18, 2008.

[11] M. Kong and P. Tian. A direct application of ant colony optimization to function optimization problem in continuous domain. In M. Dorigo et al., editors, *LNCS 4150. Proceedings of the 5th International Conference on Ant Colony Optimization and Swarm Intelligence (ANTS 2006)*, pages 324–1331, Berlin, Germany, 2006. Springer.

[12] G. Leguizamón and C. Coello. An alternative $\text{ACO}_{\mathbb{R}}$ algorithm for continuous optimization problems. In M. Dorigo et al., editors, *LNCS 6234. Proceedings of the 7th International Conference on Swarm Intelligence (ANTS 2010)*, pages 48–59, Berlin, Germany, 2010. Springer.

[13] C. Ling, S. Jie, Q. Ling, and C. Hongjian. A method for solving optimization problems in continuous space using ant colony algorithm. In M. Dorigo et al., editors, *LNCS 2463. Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002)*, pages 288–289, Berlin, Germany, 2002. Springer.

[14] M. Lozano, D. Molina, and F. Herrera. Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 2010. Forthcoming. DOI: 10.1007/s00500-010-0639-2.

[15] N. Monmarché, G. Venturini, and M. Slimane. On how Pachycondyla apicalis ants suggest a new search algorithm. *Future Generation Computer Systems*, 16(9):937–946, 2000.

[16] M. A. Montes de Oca, D. Aydın, and T. Stützle. An incremental particle swarm for large-scale optimization problems: An example of tuning-in-the-loop (re)design of optimization algorithms. *Soft Computing*, 2010. Forthcoming. DOI: 10.1007/s00500-010-0649-0.

[17] M. A. Montes de Oca and T. Stützle. Towards incremental social learning in optimization and multiagent systems. In W. Rand et al., editors, *Workshop on Evolutionary Computation and Multiagent Systems Simulation. GECCO 2008*, pages 1939–1944. ACM Press, New York, 2008.

[18] M. A. Montes de Oca, T. Stützle, K. Van den Enden, and M. Dorigo. Incremental social learning in particle swarms. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 2010. Forthcoming. DOI: 10.1109/TSMCB.2010.2055848.

[19] S. Pourtakdoust and H. Nobahari. An extension of ant colony system to continuous optimization problems. In M. Dorigo et al., editors, *LNCS 3172. Proceedings of the 4th International Conference on Ant Colony Optimization and Swarm Intelligence (ANTS 2004)*, pages 158–173, Berlin, Germany, 2004. Springer.

[20] M. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155, 1964.

[21] M. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 2009.

[22] K. Socha. ACO for continuous and mixed-variable optimization. In M. Dorigo et al., editors, *LNCS 3172. Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2004)*, pages 25–36, Berlin, Germany, 2004. Springer.

[23] K. Socha and C. Blum. An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. *Neural Computing & Applications*, 16(3):235–247, 2007.

[24] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173, 2008.

[25] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[26] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report 2005005, Nanyang Technological University, 2005.

[27] K. Tang, X. Yao, P. Suganthan, C. MacNish, Y. Chen, C. Chen, and Z. Yang. Benchmark functions for the CEC 2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007. URL: `http://nical.ustc.edu.cn/cec08ss.php`.

[28] L. Tseng and C. Chen. Multiple trajectory search for large scale global optimization. In *Proceeding of the IEEE 2008 Congress on Evolutionary Computation (CEC 2008)*, pages 3052–3059, Piscataway, NJ, 2008. IEEE Press.