An Incremental Ant Colony Algorithm with Local Search for Continuous Optimization

Tianjun Liao
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
tliao@ulb.ac.be

Marco A. Montes de Oca
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
mmontes@ulb.ac.be

Doğan Aydın
Dept. of Computer Engineering, Ege University, Izmir, Turkey
dogan.aydin@ege.edu.tr

Thomas Stützle
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
stuetzle@ulb.ac.be

Marco Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
mdorigo@ulb.ac.be

ABSTRACT

$ACO_{\mathbb{R}}$ is one of the most popular ant colony optimization algorithms for tackling continuous optimization problems. In this paper, we propose $IACO_{\mathbb{R}}$-LS, which is a variant of $ACO_{\mathbb{R}}$ that uses local search and that features a growing solution archive. 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 $IACO_{\mathbb{R}}$-LS with Mtsls1 ($IACO_{\mathbb{R}}$-Mtsls1) is not only a significant improvement over $ACO_{\mathbb{R}}$, but that it is also competitive with the state-of-the-art algorithms described in a recent special issue of the Soft Computing journal. Further experimentation with $IACO_{\mathbb{R}}$-Mtsls1 on an extended benchmark functions suite, which includes functions from both the special issue of Soft Computing and the IEEE 2005 Congress on Evolutionary Computation, demonstrates its good performance on continuous optimization problems.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization

General Terms

Algorithms

Keywords

Ant Colony Optimization, Continuous Optimization, Local Search, Automatic Parameter Tuning

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

Several algorithms based on or inspired by the ant colony optimization (ACO) metaheuristic [4] have been proposed to tackle continuous optimization problems [5, 9, 12, 14, 18]. One of the most popular ACO-based algorithms for continuous domains is $ACO_{\mathbb{R}}$ [21–23]. Recently, Leguizamón and Coello [11] proposed a variant of $ACO_{\mathbb{R}}$ that performs better than the original $ACO_{\mathbb{R}}$ on six benchmark functions. However, the results obtained with Leguizamón and Coello's variant 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 Soft Computing journal [13] (Throughout the rest of the paper, we will refer to this special issue as SOCO). The set of algorithms described in SOCO consists of differential evolution algorithms, memetic algorithms, particle swarm optimization algorithms and other types of optimization algorithms [13]. In SOCO, the differential evolution algorithm (DE) [24], the covariance matrix adaptation evolution strategy with increasing population size (G-CMA-ES) [1], and the real-coded CHC algorithm (CHC) [6] 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 $ACO_{\mathbb{R}}$ algorithm, called $IACO_{\mathbb{R}}$-LS, that is competitive with the state of the art in continuous optimization. We first present $IACO_{\mathbb{R}}$, which is an $ACO_{\mathbb{R}}$ with an extra search diversification mechanism that consists of a growing solution archive. Then, we hybridize $IACO_{\mathbb{R}}$ 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 [19], Powell's BOBYQA [20], and Lin-Yu Tseng's Mtsls1 [27]. An automatic parameter tuning procedure, Iterated F-race [2,3], is used for the configuration of the investigated algorithms. The best algorithm found after tuning, $IACO_{\mathbb{R}}$-Mtsls1, obtains results that are as good as the best of the 16 algorithms featured in SOCO. To assess the quality of $IACO_{\mathbb{R}}$-Mtsls1 and the best SOCO algorithms on problems not seen during their design phase, we compare their performance using an extended benchmark functions suite that includes functions from SOCO and the Special Session on Continuous Optimization of the IEEE 2005 Congress on

Evolutionary Computation (CEC 2005). The results show that IACO$_\mathbb{R}$-Mtsls1 can be considered to be a state-of-the-art continuous optimization algorithm.

2. THE ACO$_\mathbb{R}$ ALGORITHM

The ACO$_\mathbb{R}$ algorithm stores a set of k 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. Initially, the solution archive is filled with randomly generated solutions. The algorithm iteratively refines the solution archive by 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 according to their quality (from best to worst).

The core of the solution construction procedure is the estimation of multimodal one-dimensional probability density functions (PDF). The mechanism to do that in 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 coordinate index. The Gaussian kernel for coordinate i is:

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

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

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

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 using a Gaussian PDF with $\mu_j^i = s_j^i$, and σ_j^i equal to

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

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 $i = 1, ..., D$. An outline of ACO$_\mathbb{R}$ is given in Algorithm 1.

3. THE IACO$_\mathbb{R}$ ALGORITHM

IACO$_\mathbb{R}$ is an ACO$_\mathbb{R}$ algorithm with a solution archive whose size increases over time. This modification is based on the incremental social learning framework [15, 17]. A parameter *Growth* controls the rate at which the archive grows. Fast growth rates encourage search diversification while slow ones encourage intensification [15]. In IACO$_\mathbb{R}$ the optimization process begins with a small archive, a parameter *InitArchiveSize* defines its size. A new solution is added to it every *Growth* iterations until a maximum archive size, denoted by *MaxArchiveSize*, is reached. Each time a new solution is added, it is initialized using information from the best solution 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

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

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

IACO$_\mathbb{R}$ also features a mechanism different from the one used in the original ACO$_\mathbb{R}$ 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 as a guiding solution. 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.

We include an algorithm-level diversification mechanism for fighting stagnation. The mechanism 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.

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

```
Input: k, m, D, q, ξ, and termination criterion.
Output: The best solution found
  Initialize and evaluate k solutions
  // Sort solutions and store them in the archive
  T = Sort(S_1 ··· S_k)
  while Termination criterion is not satisfied do
    // Generate m new solutions
    for l = 1 to m do
      // Construct solution
      for i = 1 to D do
        Select Gaussian g_j^i according to weights
        Sample Gaussian g_j^i with parameters μ_j^i, σ_j^i
      end for
      Store and evaluate newly generated solution
    end for
    // Sort solutions and select the best k
    T = Best(Sort(S_1 ··· S_{k+m}), k)
  end while
```

4. IACO$_\mathbb{R}$ WITH LOCAL SEARCH

The IACO$_\mathbb{R}$-LS algorithm is a hybridization of IACO$_\mathbb{R}$ with a local search procedure. IACO$_\mathbb{R}$ provides the exploration needed to locate promising solutions and the local search procedure enables a fast convergence toward good solutions. In our experiments, we considered Powell's conjugate directions set [19], Powell's BOBYQA [20] and Lin-Yu Tseng's Mtsls1 [27] methods as local search procedures. We used the NLopt library [10] implementation of the first two methods and implemented Mtsls1 following the pseudocode found in [27].

In IACO$_\mathbb{R}$-LS, the local search procedure is called using

the best solution in the archive as initial point. The local search methods terminate after a maximum number of iterations, *MaxITER*, have been reached, or when the tolerance, that is the relative change between solutions found in two consecutive iterations, is lower than a parameter *FTOL*. 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 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. When a solution's failures counter is greater than or equal to *MaxFailures*, the local search procedure is not called again from this solution. Instead, the local search procedure is called from a random solution whose 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

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

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. IACO$_\mathbb{R}$-LS is shown in Algorithm 2. The C++ implementation of IACO$_\mathbb{R}$-LS is available in `http://iridia.ulb.ac.be/supp/IridiaSupp2011-008/`.

5. EXPERIMENTAL STUDY

Our study is carried out in two stages. First, 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 $f_{soco}*$). Second, 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) [25] (functions labeled as $f_{cec}*$).

In the first stage of the study, we used the 50- and 100-dimensional versions of the 19 SOCO functions. Functions

[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}.

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

Input: : ξ, p, *InitArchiveSize*, *Growth*, *MaxArchiveSize*, *FTOL*, *MaxITER*, *MaxFailures*, *MaxStagIter*, D and termination criterion.
Output: The best solution found

```
k = InitArchiveSize
Initialize and evaluate k solutions
while Termination criterion not satisfied do
  // Local search
  if FailedAttempts_best < MaxFailures then
    Invoke local search from S_best with parameters FTOL and MaxITER
  else
    if FailedAttempts_random < MaxFailures then
      Invoke local search from S_random with parameters FTOL and MaxITER
    end if
  end if
  if No solution improvement then
    FailedAttempts_best||random ++
  end if

  // Generate new solutions
  if rand(0,1)<p then
    for i = 1 to D do
      Select Gaussian g^i_best
      Sample Gaussian g^i_best with parameters μ^i_best, σ^i_best
    end for
    if Newly generated solution is better than S_best then
      Substitute newly generated solution for S_best
    end if
  else
    for j = 1 to k do
      for i = 1 to D do
        Select Gaussian g^i_j
        Sample Gaussian g^i_j with parameters μ^i_j, σ^i_j
      end for
      if Newly generated solution is better than S_j then
        Substitute newly generated solution for S_j
      end if
    end for
  end if

  // Archive Growth
  if current iterations are multiple of Growth & k < MaxArchiveSize then
    Initialize new solution using Eq.4
    Add new solution to the archive
    k ++
  end if
  // Restart Mechanism
  if # of iterations without improving S_best = MaxStagIter then
    Re-initialize T but keeping S_best
  end if
end while
```

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) [26]. Functions f_{soco7}-f_{soco11} were proposed at the

Table 1: Benchmark functions

ID	Name/Description	Uni./Multi.	Sep.	Ro.
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

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 [8,13]. In the second stage of our study, the 19 SOCO and 21 CEC 2005 functions on 50 dimensions were considered together. Some properties of the benchmark functions are listed in Table 1. The detailed description is available in [8, 25].

We applied the termination conditions used for SOCO and CEC 2005 were used, 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. We report error values defined as $f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$, where $\boldsymbol{x}$ is a candidate solution and $\boldsymbol{x}^*$ is the optimal solution. Error values lower than 10^{-14} (this value is referred to as *0-threshold*) are approximated to 0. Our analysis is based on either the whole solution quality distribution, or on the median and average errors.

5.1 Parameter Settings

We used Iterated F-race [2, 3] to automatically tune algorithm parameters. The 10-dimensional versions of the 19 SOCO functions were randomly sampled as training instances. A maximum of 50,000 algorithm runs were used as tuning budget for $ACO_{\mathbb{R}}$, $IACO_{\mathbb{R}}$-BOBYQA, $IACO_{\mathbb{R}}$-Powell and $IACO_{\mathbb{R}}$-Mtsls1. The number of function evaluations used in each run is equal to 50,000. The best set of parameters, for each algorithm found with this process is given in Table 2. The only parameter that we set manually was *MaxArchiveSize*, which we set to 1,000.

5.2 Experimental Results and Comparison

Figure 1 shows the distribution of median and average errors across the 19 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 0.05 α-level with a Wilcoxon test 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 objective function value lower than 10^{-14}) is given.

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 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.

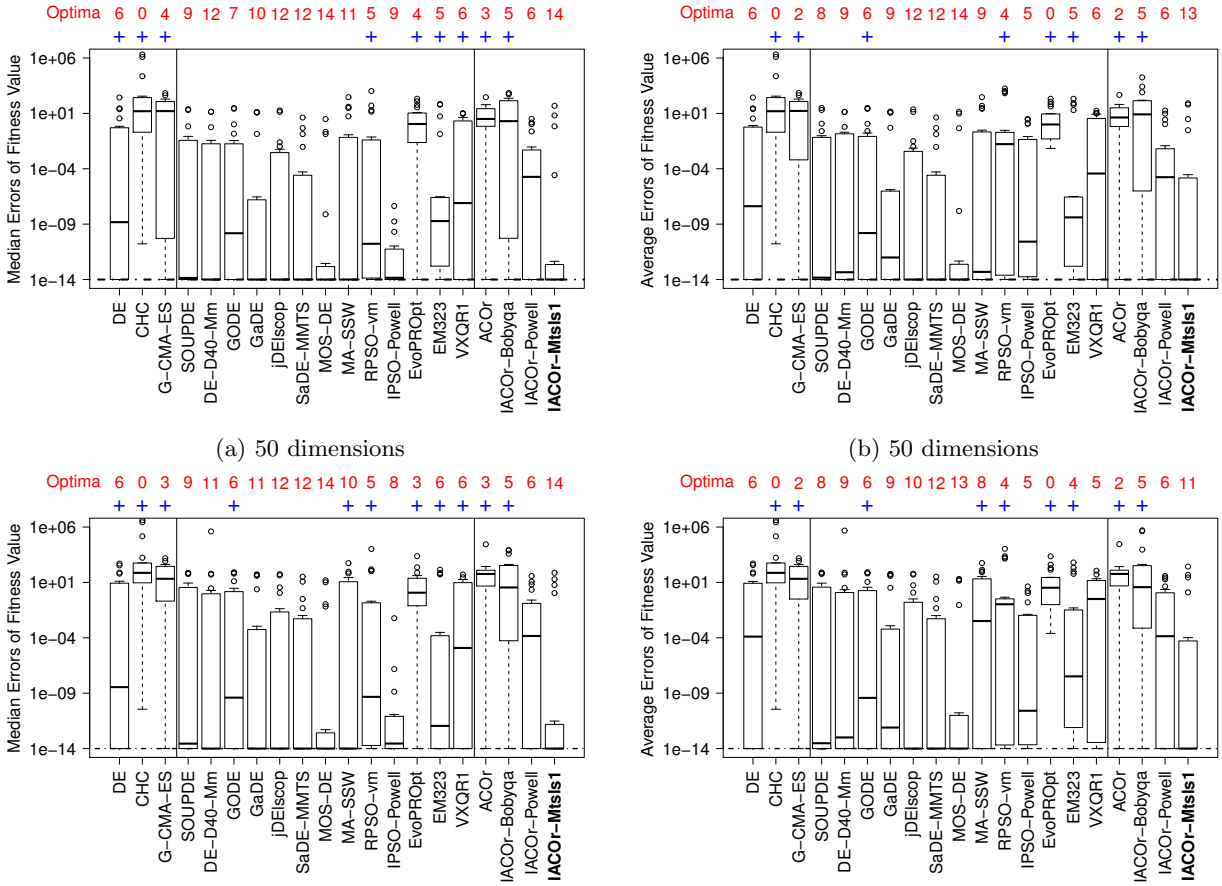
If one considers mean values, the performance of $IACO_{\mathbb{R}}$-Mtsls1 degrades slightly. This is an indication that $IACO_{\mathbb{R}}$-Mtsls1 still stagnates with some low probability. However, $IACO_{\mathbb{R}}$-Mtsls1 still outperforms G-CMA-ES, CHC, GODE, EVoPROpt, RPSO-vm, and EM323. Even 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.

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 the 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 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

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

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. The parameter *FTOL* is first transformed as 10^{FTOL} before using it in the algorithms.

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



(a) 50 dimensions

(b) 50 dimensions

(c) 100 dimensions

(d) 100 dimensions

Figure 1: The box-plots show the distribution of the median (left) and average (right) errors obtained on the 19 SOCO benchmark functions of 50 (top) and 100 (bottom) dimensions. 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 0.05 α-level detected with a Wilcoxon test between the results obtained with the indicated algorithm and those obtained with $IACO_{\mathbb{R}}$-Mtsls1. The absence of a symbol means that the difference is not significant with $IACO_{\mathbb{R}}$-Mtsls1. The numbers on top of a box-plot denotes the number of optima found by the corresponding algorithm.

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 5.1.

Table 3 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, $IACO_{\mathbb{R}}$-Mtsls1 does it as well. However, $IACO_{\mathbb{R}}$-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

Table 3: 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 functions with $D = 50$. 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 avoid repeated test on the similar functions such as f_{soco1}, f_{soco3}, f_{soco4}, f_{soco8}. At the bottom of the table, we report the number of times an algorithm found the lowest error.

	Median errors					Mean errors			
Function	G-CMA-ES	IPSO-Powell	IPSO-Mtsls1	IACO$_{\mathbb{R}}$-Mtsls1	Function	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
# of best	18	15	18	21	# of best	14	10	10	22

mean values are considered. Second, G-CMA-ES finds more best results on the CEC 2005 functions than on the SOCO functions. Overall, however, 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 IACO$_{\mathbb{R}}$-Mtsls1 and G-CMA-ES, IPSO-Powell and IPSO-Mtsls1. On the x-axis, the coordinates are the results obtained with IACO$_{\mathbb{R}}$-Mtsls1; on the y-axis, the coordinates are the results obtained with the other algorithms for each of the 40 functions. Thus, points that appear on the left part of the correlation plot correspond to functions for which IACO$_{\mathbb{R}}$-Mtsls1 has better results than the other algorithm.

Table 4 shows a detailed comparison presented in form of (win, draw, lose) according to different properties of the 40 functions used. The two-sided p-values of Wilcoxon matched-pairs signed-ranks test of IACO$_{\mathbb{R}}$-Mtsls1 with other algorithms across 40 functions are also presented. In general, IACO$_{\mathbb{R}}$-Mtsls1 performs better more often than all the other compared algorithms. IACO$_{\mathbb{R}}$-Mtsls1 wins more often against G-CMA-ES; however, G-CMA-ES performs clearly better than IACO$_{\mathbb{R}}$-Mtsls1 on rotated functions, which can be explained by the covariance matrix adaptation mechanism [7].

6. CONCLUSIONS

In this paper, we have introduced IACO$_{\mathbb{R}}$-LS, an 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 IACO$_{\mathbb{R}}$-LS. Through automatic tuning across 19 functions, IACO$_{\mathbb{R}}$-Mtsls1 proved to be superior to the other two variants.

The results of a comprehensive experimental comparison with 16 algorithms featured in a recent special issue of the

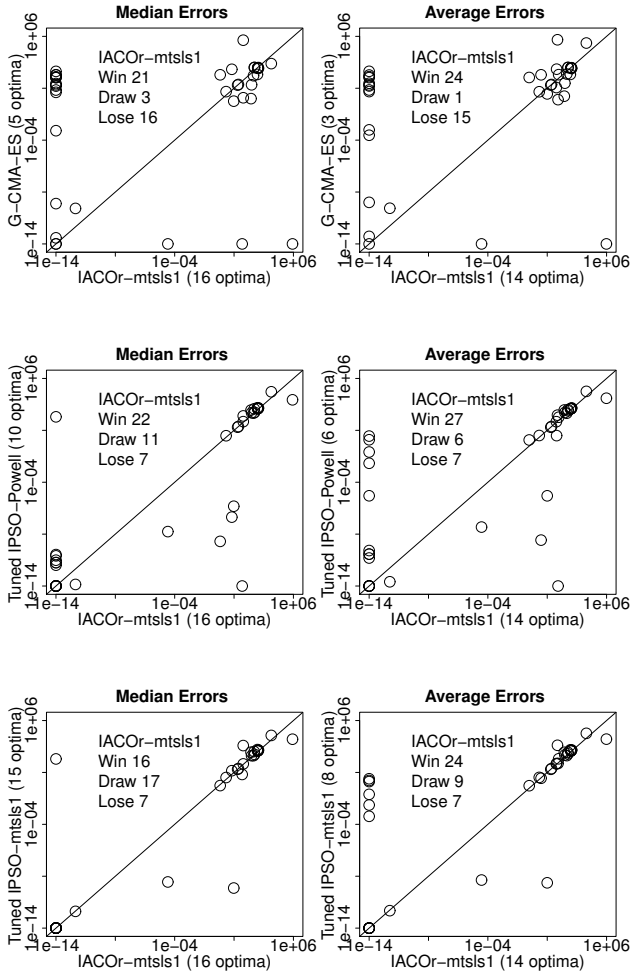


Figure 2: The correlation plot between IACO$_{\mathbb{R}}$-Mtsls1 and G-CMA-ES, IPSO-Powell and IPSO-Mtsls1 over 40 functions. Each point represents a function. The points on the left part of correlation plot illustrate that on those represented functions, IACO$_{\mathbb{R}}$-Mtsls1 obtains better results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

Soft Computing journal show that IACO$_{\mathbb{R}}$-Mtsls1 significantly outperforms the original ACO$_{\mathbb{R}}$ and that IACO$_{\mathbb{R}}$-Mtsls1 is competitive with the state of the art. We also conducted a second comparison that included 21 extra functions from the special session on continuous optimization of the IEEE 2005 Congress on Evolutionary Computation. From this additional comparison we can conclude that IACO$_{\mathbb{R}}$-Mtsls1 remains very competitive. It mainly shows slightly worse results than G-CMA-ES on functions that are rotated w.r.t. the usual coordinate system. In fact, this is maybe not surprising as G-CMA-ES is the only algorithm of the 20 compared ones that performs very well on these rotated functions. In further work we may test ACO$_{\mathbb{R}}$ in the version that includes the mechanism for adjusting for rotated functions [23] to check whether these potential improvements transfer to IACO$_{\mathbb{R}}$-Mtsls1. Nevertheless, the very good performance of IACO$_{\mathbb{R}}$-Mtsls1 on most of the Soft Computing benchmark functions is a clear indication of the high potential ACO algorithms have for this problem domain. In fact, IACO$_{\mathbb{R}}$-Mtsls1 is clearly competitive with state-of-the-art continuous optimizers.

Table 4: The comparison is conducted based on median and average errors of objective value and the results of 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 two-sided p-values of Wilcoxon matched-pairs signed-rank test of IACO$_{\mathbb{R}}$-Mtsls1 at a 0.05 α-level with other algorithms are also presented

Median Errors			
Properties of Functions	IACO$_{\mathbb{R}}$-Mtsls1 vs G-CMA-ES	IACO$_{\mathbb{R}}$-Mtsls1 vs IPSO-Powell	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	8.33E−01	6.03E−03	1.32E−02
Average Errors			
Properties of Functions	IACO$_{\mathbb{R}}$-Mtsls1 vs G-CMA-ES	IACO$_{\mathbb{R}}$-Mtsls1 vs IPSO-Powell	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)	(11, 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	4.22E−01	1.86E−03	1.66E−03

7. ACKNOWLEDGMENTS

This work was supported by the E-SWARM project, funded by an ERC Advanced Grant, and by the Meta-X project, funded by the Scientific Research Directorate of the French Community of Belgium. Thomas Stützle and Marco Dorigo acknowledge support from the Belgian F.R.S.-FNRS, of which they are a Research Associate and a Research Director, respectively.

8. REFERENCES

[1] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *Proc. of*

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 *Proc. of HM 2007*, volume 4771 of *LNCS*, pages 108–122, Germany, 2007. Springer.

[3] 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. Germany, Springer.

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

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

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

[7] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In *Proc. of 6th ICGA*, pages 57–64, San Francisco, CA, 1995. Morgan Kaufmann.

[8] 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`.

[9] 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.

[10] S. Johnson. The NLopt nonlinear-optimization package, 2008. URL: `http://ab-initio.mit.edu/wiki/index.php/NLopt`.

[11] G. Leguizamón and C. Coello. An alternative $ACO_{\mathbb{R}}$ algorithm for continuous optimization problems. In M. Dorigo et al., editors, *Proc of ANTS 2010*, volume 6234 of *LNCS*, pages 48–59, Germany, 2010. Springer.

[12] 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, *Proc. of ANTS 2002*, volume 2463 of *LNCS*, pages 288–289, Germany, 2002. Springer.

[13] M. Lozano, D. Molina, and F. Herrera. Editorial: Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 2011. In Press.

[14] 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.

[15] 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*, 41(2):368–384, 2011.

[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*, 2011. In press. 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, *ECoMASS Workshop of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 1939–1944. ACM Press, New York, 2008.

[18] S. Pourtakdoust and H. Nobahari. An extension of ant colony system to continuous optimization problems. In M. Dorigo et al., editors, *Proc. of ANTS 2004*, volume 3172 of *LNCS*, pages 158–173, Germany, 2004. Springer.

[19] 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.

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

[21] K. Socha. ACO for continuous and mixed-variable optimization. In M. Dorigo et al., editors, *Proc. of ANTS 2004*, volume 3172 of *LNCS*, pages 25–36, Germany, 2004. Springer.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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`.

[27] L. Tseng and C. Chen. Multiple trajectory search for large scale global optimization. In *Proc. of CEC 2008*, pages 3052–3059, Piscataway, NJ, 2008. IEEE Press.