Investigating Surrogate-assisted Cooperative Coevolution for Large-Scale Global Optimization

Ivanoe De Falco[a], Antonio Della Cioppa[b], Giuseppe A. Trunfio[*,c]

[a]*Institute of High Performance Computing and Networking, National Research Council of Italy, Naples, Italy*
[b]*Natural Computation Lab, DIEM, University of Salerno, Fisciano (SA), Italy*
[c]*DADU, University of Sassari, Alghero, Italy*

Abstract

In many relevant fields, real world problems involving large-scale global optimization (LSGO) in the continuous domain are becoming more and more frequent. For this reason, the last few years have seen an increasing number of researchers interested in improving optimization metaheuristics in such a way as to cope effectively with high-dimensional search domains. Among the several techniques to enhance scalability, one of the most studied is Cooperative Coevolution (CC), an effective divide-and-conquer strategy for decomposing a large-scale problem into lower-dimensional subcomponents. However, despite the progress made in the LSGO field, one of such optimizations can still require a very high number of objective function evaluations to provide a satisfactory result. Therefore, when the evaluation of a candidate solution requires complex calculations, optimizing a high number of variables can easily become a challenging task. Nonetheless, to date few studies have investigated the application of optimization metaheuristics to objective functions that are simultaneously high-dimensional and computationally significant. To address such a research issue, this article investigates a surrogate-assisted CC (SACC) optimizer, in which fitness surrogates are exploited within the low-dimensional subcomponents resulting from the problem decomposition. The developed SACC algorithm is investigated through a computational study based on a rich test-bed composed of 1000-dimensional problems. According to the results, SACC is able to significantly boost the convergence of the CC optimizer, leading in many cases to a relevant computational gain.

Key words: Large Scale Optimization, Cooperative Coevolution, Fitness metamodeling, Differential Evolution

1. Introduction

In recent years, a great deal of research efforts has been aimed at improving the effectiveness of optimization metaheuristics in addressing continuous problems characterized by several hundreds to thousands of variables. Such a field of research, which is referred to as 'large-scale global optimization' (LSGO) in the literature [38, 32], has been motivated by two main reasons.

On the one hand, there is an ever-increasing need to tackle real-world problems requiring the optimization of a high number of variables. Such applications are often characterized by lack of detailed information on the objective functions, complex fitness landscapes and constraints on computing time. In these cases, suitable metaheuristics can represent an invaluable tool for finding, within a reasonable time scale, an acceptable solution. Relevant examples of such optimization problems can be found in clustering [16, 36], data analytics [11], optimization of networks [15], engineering optimization [9], parameter estimation in systems biology [59] (e.g., genetic networks [64, 29]) and simulation of complex systems [2]. On the other hand, even search algorithms that perform satisfactorily when applied to small or medium-sized problems are characterized by a rapid decay of search efficiency as the dimensionality of the problem increases. Such an issue, typically referred to as the *curse of dimensionality*, suggested to search for suitable techniques to improve the scalability

[*]Corresponding address: DADU, University of Sassari, P.zza Duomo, 6, 07041 Alghero (SS), Italy - Phone: (+39) 079 9720427 - FAX: (+39) 079 9720420

Email addresses: ivanoe.defalco@icar.cnr.it (Ivanoe De Falco), adellacioppa@unisa.it (Antonio Della Cioppa), trunfio@uniss.it (Giuseppe A. Trunfio)

Preprint submitted to Information Sciences

Published as:
De Falco, Ivanoe, Della Cioppa, Antonio, Trunfio, Giuseppe A. (2019). Investigating surrogate-assisted cooperative coevolution for large-Scale global optimization. INFORMATION SCIENCES, vol. 482, p. 1-26, ISSN: 0020-0255, doi: 10.1016/j.ins.2019.01.009

of optimization metaheuristics. However, despite the progress made in terms of efficiency, an advanced metaheuristics algorithm still requires a significant number of objective function evaluations to find an optimal or a satisfactory near-optimal solution. To give an idea, the typical test suite adopted for the competitions on LSGO, within the IEEE Congress on Evolutionary Computation, included problems with 1000 variables and allowed 3×10^6 function evaluations [60, 61]. Unfortunately, such a high number of fitness evaluations can represent a serious limitation when the objective function must be evaluated through complex computational models (e.g., computer simulations, neural networks, etc.), as is the case in many real-world applications. In spite of this, not much attention has been given so far to the issue of using metaheuristics for optimizing high-dimensional functions that are also computationally significant.

When fitness evaluation is difficult, a typical approach adopted to decrease the computational cost of optimization consists of exploiting cheaper approximations of the objective function during the search (i.e., the so-called *surrogates* or *metamodels*) [26, 27]. However, for high-dimensional problems also the quality of fitness surrogates is plagued by the curse of dimensionality, that is, it decreases rapidly as the number of involved variables increases. For this reason, very few works address LSGO problems through metamodeling [63, 34, 58].

To address the issue of LSGO with complex objective functions, in this article we study the use of fitness surrogates in the context of a CC optimizer. The rationale behind the approach lies in the fact that, thanks to the CC technique, the surrogates can operate within the lower dimensional search spaces originated from a decomposition of the original problem [45, 23]. This allows approximating effectively the exact fitness function at the obvious cost of building a higher number of surrogates (i.e., at least one for each subproblem originated by the CC decomposition), under the assumption that there is still a saving of computing time due to the significant cost of the original fitness function.

In the literature, accurate metamodeling in conjunction with CC has been rarely used and the resulting framework has never been thoroughly investigated using LSGO problems [45, 23]. Nevertheless, quantifying the advantages provided by such a surrogate-assisted CC (SACC) is made interesting by some specific characteristics of the CC technique. For example, most optimization problems are *nonseparable*, which means that the subcomponents of a CC decomposition cannot be optimized independently [49]. However, as noted in a previous study limited to some medium-sized problems [45], the uncertainties introduced by the approximate fitness can have contrasting effects on the convergence of the search process in case of nonseparable objective functions. Moreover, it is known that given a CC optimizer there may exist an optimal size of the decomposition, dependent on both the problem and the underlying search algorithm, able to maximize the speed of convergence [43, 66]. In contrast, a lower-dimensional decomposition enables in general surrogates that are more accurate. Therefore, it is worth investigating how the adopted decomposition can affect the search efficiency of a SACC algorithm. Another peculiar aspect of the CC technique is the periodical exchange of information between subcomponents, which should be more frequent for highly nonseparable problems. In case of a SACC, with metamodeling at the subcomponent level, such interactions interfere with the accumulation of knowledge on the fitness landscape that is required to train the surrogates.

To investigate the above issues, we use and compare four common metamodeling approaches, namely Quadratic Polynomial Approximation (QPA) [22], Radial Basis Function Network (RBFN) [73, 52], Gaussian Process (GP) [13, 72, 34] and Support Vector Regression (SVR) [56], within a CC optimizer based on Differential Evolution (DE) [57].

In the article, we provide the following main contributions: (*i*) given the lack of such a study in the LSGO literature, we quantify, using a standard test-bed composed of 1000-dimensional problems, the gain of original function evaluations that can be achieved with a SACC optimizer based on advanced metamodeling and an effective search algorithm; (*ii*) we provide useful indications on the minimum cost of the objective function that makes the surrogate-assisted approach suitable for LSGO applications, where the number of metamodels to train can be significantly high; (*iii*) we investigate the effect of the relevant parameters that can affect the results.

The article is organized as follows. Section 2 provides a background on the related work and motivates, in more detail, the presented study. Section 3 describes the proposed approach, including the four variations in terms of metamodeling strategies. Then, section 4 illustrates the results of a comprehensive computational study, which includes the empirical evaluations of factors like size of decomposition and type of metamodeling. Conclusions and some hypotheses on future developments of this research are provided in section 5.

Algorithm 1: CCRG(f, d)

1 $\mathbf{P} \leftarrow$ initPopulation(d, N_{pop});
2 $\mathbf{b} \leftarrow$ initContextVector($\mathbf{P}$);
3 *fitnessEvaluations* $\leftarrow 0$;
4 **while** *fitnessEvaluations* $< N_{FE}$ **do**
5 $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_k\} \leftarrow$ randomGrouping(n, k);
6 **foreach** $\boldsymbol{G}_i \in \mathcal{G}$ **do**
7 $\mathbf{P}_i \leftarrow$ extractPopulation($\mathbf{P}$, $\mathbf{G}_i$);
8 $\langle \mathbf{P}_i, best_i, FE \rangle \leftarrow$ optimizer(f, $\mathbf{P}_i$, $\mathbf{b}$, $\mathbf{G}_i$, N_{ite});
9 $pop \leftarrow$ storePopulation($\mathbf{P}_i$, $\mathbf{G}_i$);
10 *fitnessEvaluations* $\leftarrow$ *fitnessEvaluations* + *FE*;
11 **foreach** $\boldsymbol{G}_i \in \mathcal{G}$ **do**
12 $\mathbf{b} \leftarrow$ updateContextVector($best_i$, $\mathbf{G}_i$, $\mathbf{b}$);
13 return $f(\mathbf{b})$ and $\mathbf{b}$;

2. Background and motivations

2.1. The cooperative coevolutionary optimization strategy

We consider the following minimization problem:

$$\min f = F(\mathbf{x}), \quad \mathbf{x} \in \mathbf{S} \tag{1}$$

where $\mathbf{S} \subseteq \mathbb{R}^d$ denotes a search space with a high number d of variables and $F : \mathbf{S} \rightarrow \mathbb{R}$ is a real-valued objective function. According to the CC technique [49], the d-dimensional set of search directions $G = \{1, 2, \ldots, d\}$ is first partitioned into k sets $\mathbf{G}_1 \ldots \mathbf{G}_k$, where each group $\mathbf{G}_i$ of directions defines a *subcomponent*. For example, a straightforward decomposition of the original d-dimensional search space consists of k subcomponents of the same dimension $d_k = d/k$, with the groups of directions defined as:

$$\mathbf{G}_i = \{(i-1) \times d_k + 1, \ldots, i \times d_k\} \tag{2}$$

Then, a standard iterative algorithm is applied to separately evolve candidate solutions in each subcomponent, using fixed values for the variables not included within that particular subcomponent. Since each subcomponent manages only sub-vectors of the d-dimensional candidate solutions, an exchange of information (i.e., cooperation) is required to evaluate the objective function. To this purpose, each subcomponent provides a representative individual to build a common d-dimensional *context vector* $\mathbf{b} \in \mathbb{R}^d$. Then, before its evaluation, each candidate solution belonging to a subcomponent is complemented through the appropriate elements of $\mathbf{b}$. Typically, either the current best or a randomly chosen individual is used to represent a subcomponent in the context vector [71].

When applying a CC optimizer, an important aspect to be considered is the possible epistatic interaction between variables, which can negatively affect the convergence rate [49]. To cope with such an issue in case of nonseparable problems, the typical approach consists of using equally sized subcomponents together with a dynamic grouping strategy called *Random Grouping* (RG) [74], in which the directions of the original search space are periodically re-grouped in a random way to determine the CC subcomponents. In practice, the RG strategy increases the probability of having grouped together two dependent variables during the optimization process [74, 42] and represents an effective approach when it is not possible to apply specific algorithms designed to find and group together all the interacting variables [10, 44].

An outline of a CC optimizer based on the RG strategy is shown in Algorithm 1. The first step consists of creating k groups of coordinates randomly drawn without replacement from the set $\{1, 2, \ldots, d\}$. Then, both the population, composed of N_{pop} individuals, and the context vector are randomly initialized. The optimization is organized in *cycles* (lines 5-12 of Algorithm 1). During each cycle, the optimizers are activated in a round-robin fashion for the different subcomponents (lines 6-8). Then, the context vector is updated using the current best individual of each sub-population (lines 10-11) and a new RG is executed before the next cycle (line 12). Each optimizer is executed for N_{ite} iterations at each cycle. The CC cycles terminate when the number of fitness evaluations reaches the value N_{FE}.

2.2. Surrogate-assisted LSGO in the literature

The rationale behind surrogate-assisted search algorithms consists of increasing the efficiency by building and exploiting a relatively inexpensive *surrogate model* $\hat{f}(\mathbf{x})$ of the original fitness $f(\mathbf{x})$, based on the information available from past evaluations of the objective function [26]. Typically, the model $\hat{f}$ is built using an appropriate learning algorithm, which exploits an archive of n observations $\mathcal{A} = \{\mathbf{X}, \mathbf{y}\}$, where $\mathbf{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$ are d-dimensional points of the search space and $\mathbf{y} = \{f(\mathbf{x}^{(1)}), \ldots, f(\mathbf{x}^{(n)})\}$ are the corresponding values of the fitness function obtained during or before the optimization.

Several metamodeling techniques have been proposed in the literature so far, including Polynomial Regression, GP, Artificial Neural Networks and RBFN [26, 27]. Nevertheless, for the reasons mentioned in Section 1, most of the surrogate-assisted optimization algorithms only works with relatively low-dimensional problems. Some authors used a dimensionality reduction approach based on a mapping from the original space to a lower dimensional space (e.g., the Sammon mapping used in [63, 34]). Nevertheless, such an approach necessarily involves a certain loss of information and proved suitable only for medium-scale optimization problems (e.g., $50 - 100$ variables) [63, 34]. More recently, Sun et al. in [58] addressed problems up to 500 decision variables using a surrogate-assisted competitive swarm optimizer (CSO). The approach consisted on replacing part of the expensive fitness evaluations by exploiting a fitness inheritance (FI) strategy [55], which can be seen as a local surrogate model. The authors showed that on benchmark problems up to 500 variables, the surrogate assisted CSO provided better or competitive results compared to CSO without any fitness approximation strategy. A further work on surrogate-assisted optimizers operating on high-dimensional problems was the windowed optimization approach presented by Werth et al. in [70]. The authors, exploited a preliminary static decomposition of the problems based on recognizing the structure of variable interactions. In every iteration, the optimization is conducted on a sliding window inclusing only a subset of the variables. Such an approach proved effective on a preliminary investigation based on a five optimization problems up to 1000 decision variables.

A natural approach to cope with the curse of dimensionality when training surrogates is to operate within the lower dimensional search spaces of a CC decomposition. However, while the research in LSGO led to several applications of effective CC optimizers [74, 43, 68, 39], the joint use of the CC approach together with fitness metamodeling is relatively rare in literature, especially when the application to LSGO problems is concerned.

An earlier work based on CC optimization assisted by approximate models was conducted in 2001 by Nair and Keane in [40], where the authors studied the design of a large space structure to achieve passive vibration suppression. The optimization problem of size 40 was decomposed into 5 sub-problems of 8 variables and the approach proved effective when compared with a traditional GA.

Later, in [45], Ong and co-authors used a surrogate-assisted CC based on a real-coded GA and a RBFN [6] to approximate the exact fitness. The main aim of the study was to investigate the combined effect of non-separability and fitness metamodeling. Unfortunately, the approach was not applied to LSGO, since the investigation was based on only two objective functions, with 20 and 40 variables respectively. Moreover, the surrogate-assisted CC used in [45] did not use the RG approach. Nevertheless, such a surrogate-assisted CC-GA proved promising and, surprisingly, the preliminary experimentation suggested that the fitness noise due to the use of surrogates could compensate, to some extent, the negative effect of epistasis. In other words, the advantages provided by the metamodeling could apparently overcome the loss of efficiency related to the decomposition of a nonseparable problem. Indeed, such an advantage could be related with what was noted since the first application of the CC approach for some nonseparable problems [49], namely that a random selection of the representative individual in each subcomponent can improve the search efficiency. However, on the other hand, overly inaccurate surrogates can potentially mislead or slow down the optimization process. Interestingly, the simultaneous 'curse and blessing' of uncertainty due to surrogates is also a known effect of approximation errors in the case of standard (i.e. non-CC) evolutionary search [47, 69].

More recently, Goh and co-authors described in [23] a surrogate-assisted memetic CC algorithm for constrained optimization problems. In the proposed approach, the exact objective function is used within each subcomponent when applying a GA adopted as underlying optimizer. During the process, an archive of already evaluated individuals is maintained in each subcomponent. After each CC cycle, a Local Search (LS) phase takes place, within each subcomponent and for each individual, by exploiting fitness surrogates built on the basis of the current archive of exact evaluations. Although the proposed algorithm has proved effective in some constraint-optimization test functions, it was not investigated in high-dimensional problems.

Another relevant study is [24], where the authors investigated a simple CC optimizer based on a standard GA and endowed with the RG strategy. In such an algorithm, a fixed proportion of the offspring at each generation was evaluated

through a FI strategy [55], which can be viewed as an elementary metamodeling. The results on four typical benchmark functions, up to 1000 variables, showed a significant gain of efficiency due to the adopted FI. Later, in [25] the same FI strategy was used within two CC algorithms based on more sophisticated optimizers. In this case, the results suggested no evidence of overall advantages provided by the FI strategy. The authors concluded that, likely, the adopted surrogate-assisted approach was not able to add any significant improvement to algorithms already including several sophisticated mechanisms that boost their performance. Actually, FI is significantly inexpensive on the one hand and potentially affected by high inaccuracy on the other hand. For this reason, in case of complex objective functions it is meaningful to consider also more sophisticated metamodeling approaches, providing a better trade-off between accuracy and computational cost. Nevertheless, when accurate metamodels were used in conjunction with CC strategies, the authors did not investigate thoroughly the effectiveness of their approach for addressing LSGO problems [45, 23, 65]. Moreover, in [25] the authors considered the efficiency of their surrogate-assisted CC at the end of 5×10^6 exact fitness evaluations, which can be unsuitable in case of objective functions involving complex calculations. Indeed, it would be also relevant to quantify the gain of efficiency provided by the surrogate-assisted approach for a much lower number of original fitness evaluations. Also, considering the very high number of surrogates trained during the optimization, it would be important to provide indications on the minimum cost of the objective function that makes convenient such an approach.

Besides the research issues highlighted above, it is worth noting that every update of the context vector $\mathbf{b}$, in case of nonseparable problems, inevitably interferes with the maintenance of a subcomponent-level training archive of evaluated individuals. In fact, after each update of $\mathbf{b}$ the exact fitness to be attributed to the elements of the archive should be re-evaluated, making the support of metamodeling less efficient. In practice, this may imply evaluating several new individuals with the original fitness up to the minimum number of training patterns that allows building a reliable surrogate. Given that a random rearrangement of the decomposition implies an update of the context vector, such an issue is more significant when a high frequency of RG is used, as suggested in [42] for improving the convergence. In practice, on the one hand a more frequent RG can be beneficial in a CC algorithm, while on the other hand it can cause a loss of efficiency in the SACC algorithm.

Furthermore, while the accuracy of surrogates benefits from a lower size of subcomponents, the same does not necessarily hold for the search efficiency of the used optimizer. In fact, the value of d_k in Equation 2 can have a significant impact on the convergence during the optimization process. This has been shown in [43] for separable problems, where the authors highlighted that there is often an optimal value of d_k that maximizes the performance of a given optimizer. A further study, also including nonseparable problems, suggested that in most cases there is a range of optimal subcomponent sizes, depending on both the problem and the parameters adopted for the underlying optimizer (e.g., number of evolved individuals) [66]. As a result, while a surrogate-assisted CC should be based on a low-dimensional decomposition to promote accurate approximations of the original fitness, the same decomposition could introduce a loss of search efficiency compared with a higher-dimensional one.

In the following, all the issues outlined above will be empirically investigated using a SACC optimizer based on four different metamodeling techniques, chosen among the most popular in the literature.

3. The adopted approach

The idea investigated in this study is illustrated in Figure 1 for an optimization problem defined in a bi-dimensional search space (we used the function of problem f_6 listed in the following Table 1). Clearly, the problem was decomposed into two subcomponents, each including one of the two variables. In each of the two subcomponents, the proposed search algorithm exploits mono-dimensional surrogates to replace most of the required exact fitness evaluations. Following Algorithm 1, the subspaces covered by the two subcomponents are updated during the search, until the predetermined stop condition occurs. At the same time, the approximations of the mono-dimensional fitness landscapes are re-built accordingly.

Following a frequent choice in LSGO past works [75, 31, 66], we selected DE [57] as the evolutionary optimizer within each subcomponent. The particular DE version adopted in the present study is JADE [76], an adaptive variation in which: (*i*) specific F_i (a DE parameter used in mutation [57]) and crossover rate CR_i parameters are associated to each individual in the population; (*ii*) F_i and CR_i are randomly generated at each DE iteration from Cauchy and normal distributions, respectively; (*iii*) the means μ_{CR} and μ_F of the distributions are updated according to the historical record of success of the generated parameters. Moreover, JADE implements a mutation strategy called 'DE/current-to- *p*best', in which one of the vectors involved in every mutation is randomly selected among the $100 \cdot p\%$ best individuals, being $p \in (0, 1]$ a further

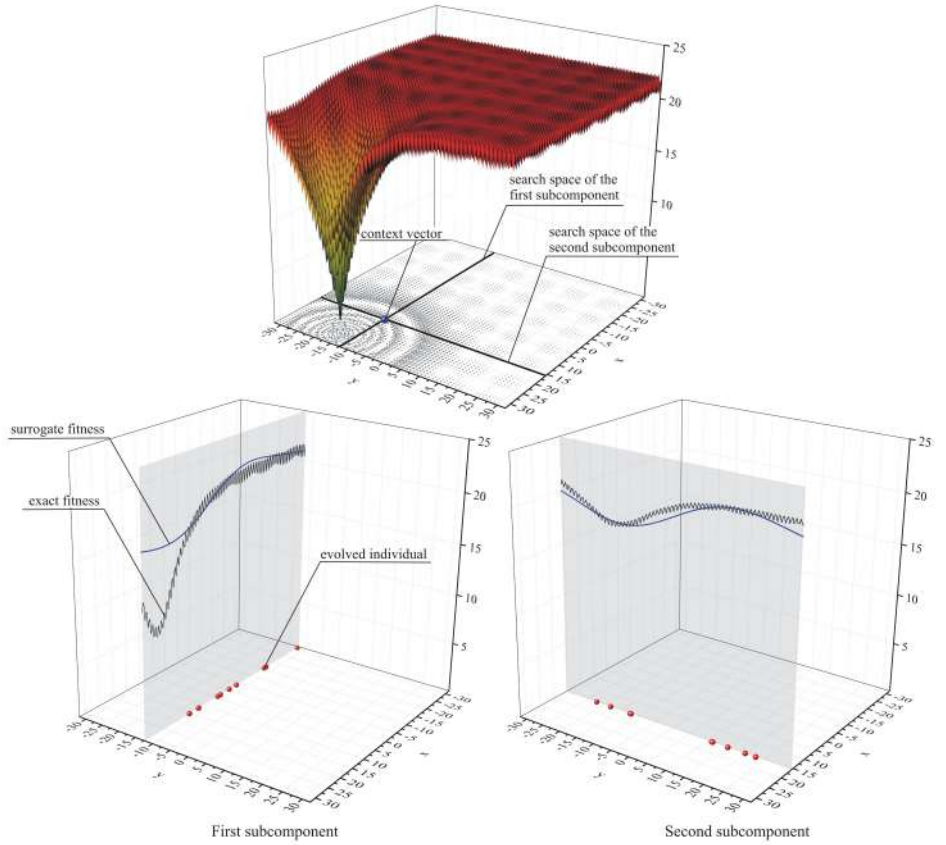

Figure 1: Exemplification of the proposed approach for an optimization problem defined in a bi-dimensional search space. In each of the two subcomponents, the algorithm exploits mono-dimensional surrogates.

parameter. According to the results reported in [76], JADE showed an excellent optimization performance on a rich suite of benchmark problems. Moreover, in [66] a CC algorithm based on JADE proved already very effective in tacking LSGO problems.

The developed SACCJADE operates according to Algorithm 1, which calls at line 8 the optimizer outlined in Algorithm 2. The latter, following an *online learning* approach [26], builds and exploits an approximate model of the objective function in order to attribute a surrogate fitness to most of the offspring produced within the CC cycle.

At each activation of the subcomponent's optimizer, the archive $\mathcal{A}$ mentioned in section 2.2 is empty (see line 1 of Algorithm 2). Then, to collect enough patterns on the current fitness landscape for building a first surrogate, all the initial individuals are evaluated with the exact fitness (lines 2-4 of Algorithm 2) and the local counter *FE* is updated accordingly (line 5). Subsequently, every evaluation of an individual with the original fitness function enriches $\mathcal{A}$ with a new pattern. At each generation, a first iteration on the parent population $\mathbf{P}$ generates an offspring population $\mathbf{C}$ according to the standard JADE procedure (line 7 of Algorithm 2) [76]. Then, at line 8, the function *evaluateOffspring* associates a fitness value $\mathbf{y}[i]$ to each newly generated vector $\mathbf{C}[i]$. However, as shown in Algorithm 3, this is done by building and using an approximate model when a sufficient number of patterns are collected.

In this study, we investigate the use of both local and global surrogates, with different computational costs and efficiency. Therefore, Algorithm 3 includes a parameter *typeOfSurrogate*, which allows choosing between a local and a global fitness model. It is worth noting that the selection of a suitable surrogate, among the different types available, should account for the characteristics of the fitness landscape. To such purpose, in the recent literature, researchers started studying strategies for automatically choosing the most appropriate surrogate in case of black-box functions [1]. Such approaches, not investigated in the present article, could provide the parameter *typeOfSurrogate* of Algorithm 3 as well as

Algorithm 2: optimizer(f, d_k, $\mathbf{P}$, $\mathbf{b}$, $\mathbf{G}_i$, N_{ite})

Used parameters: *typeOfSurrogate*, JADE parameters

1 Empty the archive $\mathcal{A}$ of past evaluations;

`// Evaluate population with the exact fitness`

2 **for** $i \leftarrow 1$ **to** $|\mathbf{P}|$ **do**

3 $\quad \boldsymbol{\phi}[i] \leftarrow f(\mathbf{P}[i])$;

4 $\quad$ Insert $\mathbf{P}[i]$ and $\boldsymbol{\phi}[i]$ in $\mathcal{A}$;

5 $FE \leftarrow |\mathbf{P}|$;

6 **for** $g \leftarrow 1$ **to** N_{ite} **do**

7 $\quad$ Using $\langle \mathbf{P}, \boldsymbol{\phi} \rangle$, generate offspring population $\mathbf{C}$ according to JADE;

$\quad$ `/* Evaluate the fitness` $\mathbf{y}[i]$ `of each newly generated individual` $\mathbf{C}[i]$ `*/`

8 $\quad \langle \mathcal{A}, \mathbf{y}, \textbf{hasExactFitness}, FE \rangle \leftarrow$ evaluateOffspring($\mathbf{C}$, $\mathcal{A}$, FE, *typeOfSurrogate*);

$\quad$ `// Ensure the best individual has exact fitness`

9 $\quad j_b \leftarrow$ index of the best individual in $\mathbf{C}$;

10 $\quad$ **while** **hasExactFitness**$[j_b]$ =**false do**

11 $\quad\quad \mathbf{y}[j_b] \leftarrow f(\mathbf{C}[j_b])$;

12 $\quad\quad$ Insert $\mathbf{C}[j_b]$ and $\mathbf{y}[j_b]$ in $\mathcal{A}$;

13 $\quad\quad FE \leftarrow FE + 1$;

14 $\quad\quad$ **hasExactFitness**$[j_b]$ =**true**;

15 $\quad\quad j_b \leftarrow$ index of the best individual in $\mathbf{C}$;

16 $\quad$ Use $\langle \mathbf{C}, \mathbf{y} \rangle$ to replace parents in $\langle \mathbf{P}, \boldsymbol{\phi} \rangle$ as in JADE;

17 $\quad$ Adapt DE parameters as in JADE;

18 **return** ***P***, *indexOfBestIndividual*($\boldsymbol{\phi}$), FE;

the specific parameters of the selected metamodeling.

More in detail, in case of global model, as soon as the archive reaches the minimum number of patterns n_r, a single fitness surrogate is built for evaluating the entire offspring (lines 4-6 of Algorithm 3). Note that when $\mathcal{A}$ contains enough elements, a new model is built at each JADE iteration with the aim of adapting the fitness model to the evolution of the population, that is to obtain at least an accurate approximation of the fitness landscape in the area covered by the current population. According to lines 7-14 of Algorithm 3, if the surrogate is not ready, a newly generated individual $\mathbf{x}$ is evaluated with the original fitness $f(\mathbf{x})$; otherwise, it is associated to an approximate fitness value $\hat{f}(\mathbf{x})$.

In case of local model, if the archive contains at least n_r patterns, a fitness surrogate is built on the fly for evaluating each candidate vector. Otherwise, the original fitness is used and the archive is consequently updated (see lines 16-25 of Algorithm 3).

As discussed in [26], the original fitness must be used during the DE iterations to avoid convergence misleading due to the differences between the surrogate and the real objective function. Therefore, at lines 9-15 of Algorithm 2 we ensure that the original fitness is attributed to the best individual generated at each iteration. Note that this requires to iterate since an individual initially classified as the best in terms of approximate fitness can then turn out to be worse than others, when evaluated exactly. Nevertheless, the required new exact evaluations contribute to the progressive improvement of the metamodel for the next iterations.

At line 16 of Algorithm 2, each element of the current population is replaced by the corresponding offspring with better fitness. Moreover, at line 17 the DE parameter adaptation is performed, as in the original JADE algorithm.

3.1. Adopted metamodeling techniques

We experimented one local meta-model, namely QPA, which, in most cases, can represent the best nonlinear trade-off between simplicity and efficiency. Moreover, we studied three commonly adopted global metamodels, namely GP, RBFN and SVR.

In the adopted QPA model, we estimate the fitness of an individual $\mathbf{x}$ as $\hat{f}(\mathbf{x}) = \boldsymbol{\beta}^T \tilde{\mathbf{x}}$, where $\boldsymbol{\beta}$ is the vector collecting the $n_p = (d_k + 1)(d_k + 2)/2$ model coefficients and the n_p-dimensional vector $\tilde{\mathbf{x}}$ is defined from $\mathbf{x}$ according to the second-

Algorithm 3: evaluateOffspring($\mathbf{C}$, $\mathcal{A}$, FE, *typeOfSurrogate*)

```
if typeOfSurrogate=globalModel then
    // Global model covering the current archive
    surrogateTrained← false;
    for i ← 1 to |C| do
        if surrogateTrained=false and |A| ≥ n_r then
            f̂ ← trainSurrogateFitness(A);
            surrogateTrained← true;
        if surrogateTrained=false then
            y[i] ← f(C[i]);
            hasExactFitness[i] =true;
            FE← FE +1;
            Insert C[i] and y[i] in A;
        else
            y[i] ← f̂(C[i]);
            hasExactFitness[i] =false;
else
    // Local model around C[i]
    for i ← 1 to |C| do
        if |A| ≥ n_r then
            f̂ ← trainSurrogateFitness(A, C[i]);
            y[i] ← f̂(C[i]);
            hasExactFitness[i] =false;
        else
            y[i] ← f(C[i]);
            hasExactFitness[i] =true;
            FE← FE +1;
            Insert C[i] and y[i] in A;
return A, y, hasExactFitness, FE;
```

degree polynomial terms. We estimate the coefficients β_i using the least square method, which requires at least $n_r \geq n_p$ elements of the archive $\mathcal{A}$. An advantage of QPA is that it can often smooth out enough the fitness landscape while well describing the global trend. Moreover, the method is computationally efficient in case of low-dimensional search spaces. However, the main drawback of a QPA metamodel lies in its inability of describing multimodal or non-symmetric fitness landscapes. To mitigate such an issue, we use an ad-hoc local QPA interpolant for each individual $\mathbf{x}$ to be evaluated. To this purpose, we solve a least square problem for the n_p nearest neighbours of $\mathbf{x}$ in $\mathcal{A}$.

The RBFN fitness surrogate is expressed as the following linear combination:

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{n_c} w_j \, \exp\left(-(\mathbf{x} - \mathbf{c}_j)^T \, \mathbf{R}_j \, (\mathbf{x} - \mathbf{c}_j)\right) \tag{3}$$

where $w_j \in \mathbb{R}$ are scalar weights, the n_c *centres* $\mathbf{c}_j \in \mathbb{R}^{d_k}$ are representative of the n_r data points in the archive $\mathcal{A}$ used for the RBFN training, and $\mathbf{R}_j$ are diagonal matrices, introducing a parameter σ_{ij} for each basis function and direction as suggested in [53]:

$$\mathbf{R}_j = diag\left(\frac{1}{2\sigma_{1j}^2}, \ldots, \frac{1}{2\sigma_{d_k j}^2}\right) \tag{4}$$

The above RBFN model depends on the vector of $n_c\,(1 + 2\,d_k)$ parameters $\mathbf{u} = \{\mathbf{w}, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_{n_c}, \mathbf{c}_1, \ldots, \mathbf{c}_{n_c}\}$, where $\mathbf{w}$ collects the weights w_i and each vector $\boldsymbol{\sigma}_j$ contains the scaling factors of the basis function centred in $\mathbf{c}_j$. The approach

adopted in this study for estimating $\mathbf{u}$ consists of using the three-phase learning described in [53]. More in detail, we first select the number $n_c < n_r$ of centres. Then, a few iterations of a k-means clustering algorithm is used to partition the archive $\mathcal{A}$ into n_c clusters, whose centroids are the initial values of $\mathbf{c}_i$. Subsequently, the values σ_{ij} are simply initialized to the variance of cluster centres $\mathbf{c}_i$ and the weights $\mathbf{w}$ are randomly initialized in $[-1, 1]$. In the third and final phase, we adjust all the parameters in $\mathbf{u}$ in order to minimize the error $E = \sum_{q=1}^{n_r}(y^{(q)} - \hat{f}(\mathbf{x}^{(q)}))^2/2$. This is done by a fixed number of a gradient descent procedure in which $\mathbf{u}$ is iteratively moved in the direction of the negative gradient $-\nabla E$ using a small learning rate η. According to some preliminary experiments, with the small-sized subcomponents used in this study (e.g., $d_k \leq 25$), a relatively small number of iterations of the above procedure is sufficient to find a suitable set of parameters.

The GP surrogate is computed using the training set $\mathcal{A} = \{\mathbf{X}, \mathbf{y}\}$ introduced in section 2.2, where the vector $\mathbf{y}$ of n_r fitness observations is considered as a sample of a multivariate Gaussian distribution with joint probability density $p(\mathbf{y} \,|\, \mathbf{X})$. When a new test point $\mathbf{x}$ is added (i.e., the point in which the value of f should be estimated), also the resulting vector $\{\mathbf{y}, f(\mathbf{x})\}$ is considered as a sample of the Gaussian joint probability density $p(\{\mathbf{y}, f(\mathbf{x})\} \,|\, \{\mathbf{X}, \mathbf{x}\})$. Using the Bayes' rule, the expected value of $f(\mathbf{x})$ can be easily expressed in terms of the inverse of the $n_r \times n_r$ matrix $\mathbf{C}(\mathbf{X}, \mathbf{X})$ of covariances between the n_r training points. As for the covariance function, we adopted the following assumtion proposed in [5]:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \theta_1 \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_{p,i} - x_{q,i})^2}{r_i^2}\right) + \theta_2 + \delta_{pq}\theta_3 \tag{5}$$

where the first term is scaled by the hyperparameter θ_1 and decreases exponentially with the distance between $\mathbf{x}_p$ and $\mathbf{x}_q$. The scale parameters r_i can attribute a different importance to the distances computed along different directions. In practice, if the distance between input points is small compared to the length scales r_i, the first term is close to θ_1; with increasing distance it exponentially decays to zero. The parameter θ_2 represents a small offset from zero and the parameter θ_3 adds a further small noise to the diagonal element of the covariance matrix (δ_{pq} is the Kronecker's delta). The latter terms improve the numerical stability of the GP modeling [51]. Equation (5) depends on a (d_k+3)-dimensional vector of hyperparameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, \theta_3, r_1, r_2, \ldots, r_{d_k}\}$. Before using the GP model, the hyperparameters $\boldsymbol{\theta}$ are optimized in such a way that the log-likelihood $\log(p(\mathbf{y} \,|\, \mathbf{X}, \boldsymbol{\theta}))$ is maximal. Since the model derivatives with respect to the hyperparameters can be easily computed [51], in the line of principle, any gradient based optimization algorithm can be used to obtain a suitable value of $\boldsymbol{\theta}$. In the present application, the GP-based surrogates were implemented using the C++ library *libgp* proposed in [3]. Moreover, the log-likelihood maximization was performed using a box-constrained optimizer included in the machine learning toolkit *dlib* [30], which exploits an implementation of the L-BFGS quasi-Newton method [7].

For the SVR-based fitness surrogates we use a Gaussian radial basis kernel and ϵ-insensitive loss function as described in [67, 56]. Besides the kernel bandwidth σ, such a model depends on two more hyperparameters, namely ϵ and C. The first controls the width of the so-called insensitive zone, that is, the loss function is defined in such a way that no penalty is associated to points predicted within a distance ϵ from the target value. Higher values of ϵ promote the use of fewer support vectors and less complex models. The value of C affects the penalty attributed to samples with errors beyond the ϵ value. Clearly, all the three hyper-parameters can significantly affect the accuracy and generalization ability of the SVR model. However, given the high number of surrogates required by our SACC algorithm, finding the SVR hyperparameters trough complex search procedures would be infeasible. Therefore, we adopted a fixed value for ϵ and a simple grid search procedure for the remaining two hyperparameters. In terms of implementation, we used the SVR model included in *dlib* [30], together with the efficient Sequential Minimal Optimization (SMO) training procedure [48].

For RBFN, GP and SVR surrogates, a single fitness metamodel is trained with the aim of approximating at least the area of the fitness landscape corresponding to the current population. In other words, within each subcomponent and at each generation, we use a surrogate based on an archive $\mathcal{A}$ containing the n_r most recently evaluated individuals.

4. Computational study

In this section, we investigate the different variations of the SACCJADE optimizer described above. We first illustrate the adopted experimental setup, including a detailed specification of the metamodels learning parameters. Then, we present and discuss several computational experiments. In the following, we denote as CCJADE our CC version of JADE

Table 1: The eleven base benchmark functions.

Function	Description	Range	Optimum	Characteristics
f_1	Shifted Sphere Function	$[100, 100]^d$	−450	unimodal, shifted, separable
f_2	Shifted Schwefel's Problem 2.21	$[100, 100]^d$	−450	unimodal, shifted, nonseparable
f_3	Shifted Rosenbrock's Function	$[100, 100]^d$	390	multimodal, shifted, nonseparable
f_4	Shifted Rastrigin's Function	$[5, 5]^d$	−330	multimodal, shifted, separable
f_5	Shifted Griewank's Function	$[600, 600]^d$	−180	multimodal, shifted, nonseparable
f_6	Shifted Ackley's Function	$[32, 32]^d$	−140	multimodal, shifted, separable
f_7	Shifted Schwefel's Problem 2.22	$[10, 10]^d$	0	unimodal, shifted, separable
f_8	Shifted Schwefel's Problem 1.2	$[65536, 65536]^d$	0	unimodal, shifted, nonseparable
f_9	Shifted Extended Schaffer	$[100, 100]^d$	0	unimodal, shifted, nonseparable
f_{10}	Shifted Bohachevsky	$[15, 15]^d$	0	unimodal, shifted, nonseparable
f_{11}	Shifted Schaffer	$[100, 100]^d$	0	unimodal, shifted, nonseparable

Table 2: The eight hybrid composition functions. Note that 'NS' is for 'not shifted'.

Function	Description	Range	Optimum
f_{12}	Hybrid NS f_9 and f_1 (25%, 75%)	$[100, 100]^d$	0
f_{13}	Hybrid NS f_9 and f_3 (25%, 75%)	$[100, 100]^d$	0
f_{14}	Hybrid NS f_9 and f_4 (25%, 75%)	$[5, 5]^d$	0
f_{15}	Hybrid NS f_{10} and NS f_7 (25%, 75%)	$[10, 10]^d$	0
f_{16}	Hybrid NS f_9 and f_1 (50%, 50%)	$[100, 100]^d$	0
f_{17}	Hybrid NS f_9 and f_3 (75%, 25%)	$[100, 100]^d$	0
f_{18}	Hybrid NS f_9 and f_4 (75%, 25%)	$[5, 5]^d$	0
f_{19}	Hybrid NS f_{10} and NS f_7 (75%, 25%)	$[10, 10]^d$	0

not relying on fitness metamodeling[1].

4.1. Experimental setup

Several large-scale optimization benchmark suites have been proposed in the literature [41, 32]. Typically, such suites includes some base test functions that are combined in different ways for increasing the number of optimization problems. In this study the computational experiments were conducted on the testing suite described in [35], which includes a greater variety of base functions when compared with other set of LSGO benchmark problems [41, 32]. The adopted testbed is composed of the 19 real-valued functions listed in Tables 1 and 2. The first 6 problems were originally proposed for the 'Special Session and Competition on Large Scale Global Optimization' held at the CEC 2008 Congress [62]. A detailed description of the next 5 functions can be found in [21]. Finally, the last 8 functions, which were introduced in [35], are obtained by summing two functions belonging to the base set $f_1 - f_{11}$. However, before summing the two contributions, the set of variables is split into two parts, each of which is attributed to a base function. In Table 2 we report the split ratios used for attributing the variables. For example, $(25\%, 75\%)$ means that 25% of the variables contribute to the first function and the remaining 75% to the second function. As can be seen in Table 1, the testing suite includes functions with different features, such as unimodal or multimodal, separable or nonseparable. Moreover, some of the functions have a global optimum point $\mathbf{x}^*$ that is shifted by a different amount in each dimension. We conducted the numerical experiments on the above test functions using a search space of $d = 1000$ variables. The error value defined as $|f(\mathbf{x}) - f(\mathbf{x}^*)|$, where $f(\mathbf{x}^*)$ is the global optimum, was adopted as a performance metric.

Typically, the maximum number of objective function evaluations adopted in the literature for this test set was 5×10^6 (e.g., [35, 31, 32]). In particular, recent results achieved by several advanced optimization metaheuristics with such a stopping criterion can be found on [32]. However, since we are interested in improving the performance of the optimizer for functions requiring nontrivial computation, we compared SACCJADE and CCJADE stopping the optimization after 5×10^5 fitness evaluations. For each function, we collected some relevant statistics from 50 independent runs.

[1]The C++ source code of the SACCJADE implementations used in the computational study can be downloaded from the following url: https://github.com/lsgo-metaheuristics/saccde

In accordance with the literature on benchmarking surrogate-assisted evolutionary optimizers [33, 34, 77, 58], the comparisons with non-surrogate-assisted approaches were carried out by adopting a predefined number of original fitness evaluations (i.e. not considering the evaluations of surrogates). Clearly, the rationale is that the benchmark function represents a real problem in which the evaluation of the exact fitness is significantly more expensive than the evaluation of the surrogates. Nevertheless, given that in our LSGO case the number of required function evaluations is necessarily large, in section 4.7 we also show the computing times related to surrogates for a test case in order to quantify the advantages as a function of the cost of the exact fitness function.

The two JADE parameters referred to as c and p in [76], and used in Algorithm 2, were both set to 0.1. Instead, the number of JADE iterations per cycle and subcomponent (i.e., N_{ite} in Algorithm 1) was the object of a specific investigation described in the following. The same applies for the number k of subcomponents, while the sizes N_{pop} of evolved population were selected considering the dimensions d_k of the resulting subcomponents.

As suggested in [5], before training, we normalized the patterns to have the objective function values in the interval $[0,\ 1]$ and the decision variables in $[-1,\ 1]$.

For the QPA local model we always used the minimum number of patterns required by the least square approach, according to the size d_k of the subcomponents. Since the latter value is relatively low in our study, RBFN, GP and SVR could provide good fitness surrogates by setting the minimum value n_r of the training data to the number of individuals in the population.

In [5] the sizes of GP training sets ranged from 15 to 120, for search spaces from 2 to 32 variables. Also, in [5, 34], too much training data used for GP did not bring significant benefits in terms of quality of the fitness estimate, while greatly increasing the computing time. Therefore, given that we considered values of $d_k \leq 25$, for the GP surrogates we used a maximum number of 80 training samples, by choosing the ones most recently generated. In addition, for the GP training, the L-BFGS algorithm was executed with error tolerance $1.0E{-}08$ for finding the hyperparameters in the following bounds used in [5]: $\theta_1 \in [10^{-3}, 1],\ \theta_2 \in [10^{-3}, 1],\ \theta_3 \in [10^{-9}, 10^{-2}],\ r_i \in [10^{-2}, 10]$.

For the RBFN, we used $n_c = n_r/5$ and we did not set a maximum number of patterns. Moreover, each training phase was carried out with 30 iterations of gradient descent with learning rate of 0.1. Such settings proved effective in some preliminary tests and slightly different values (e.g., increasing the number of iterations or changing the number of centers) did not provide significantly better results.

As for the SVR training, after some preliminary experiments, we set $\epsilon = 10^{-8}$, $\sigma \in \{1, 100\}$ and $C \in \{1, 100\}$. The search for the best σ and C was performed on a $5{\times}5$ grid. In particular, first the available archive of samples was randomly split into a training set (80% of the samples) and a test set (20% of the samples). Then, among the SVR models obtained trough SMO trainings for each couple $\langle \sigma, C \rangle$ we selected the one leading to the lowest error on the test set. Also for building the SVR model we used the entire archive $\mathcal{A}$ at each iteration.

In Algorithm 2, we ensure that the best individual at each JADE generation, and within each subcomponent, is evaluated with the exact fitness function. However, we have also preliminary evaluated the influence of a different number N_b of exact fitness evaluations per CC cycle. The results showed that increasing N_b above a relatively low value of 1 or 2 individuals determines a deterioration of the average SACCJADE performance. Indeed, a high sensitivity to such a parameter was expected, given that the resulting number of exact evaluations at each CC cycle is obtained by multiplying N_b by both the number k of subcomponents and the number N_{ite} of JADE iterations per cycle.

Another preliminary investigation concerned the availability, for the GP-based surrogate, of the analytical expression for estimating the variance, which is a statistically sound proxy of the uncertainty in its fitness estimation [13, 5, 34]. In fact, a high variance predicted at a point may indicate an area of the fitness landscape still undersampled with respect to the need of building an accurate GP. Our preliminary experiments showed that including an additional exact fitness evaluation for the individual with the highest predicted variance could help to improve GP-based surrogates.

Therefore, in the following numerical experiments, we used Algorithm 2 (i.e., $N_b = 1$) for the RBFN, SVR and QPA metamodeling. However, we used a slightly modified version for the GP-based SACCJADE, in which also the individual with the highest variance is re-evaluated with the exact fitness.

4.2. Preliminary assessment of the CCJADE search efficiency

As argued by Hameed et al., discussing the case of fitness inheritance used in conjunction with CC [25], it can be important to evaluate a surrogate-assisted technique within an efficient optimizer. In fact, the enhancement of a straightforward evolutionary algorithm is relatively easy to achieve [24]. In contrast, the use of metamodeling within a CC

Table 3: Average results obtained by CCJADE and by some state-of-the art optimization algorithms on the adopted test functions.

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7
MOS-SOCO2011	0.00E+00	5.88E-01	7.09E+01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
jDElscop	0.00E+00	2.46E+01	8.51E+02	2.39E-01	0.00E+00	1.16E-12	0.00E+00
GaDE	0.00E+00	5.46E+01	9.47E+02	3.79E-02	5.91E-04	2.55E-14	0.00E+00
DECC-G	3.26E-06	1.31E+03	1.09E+00	2.16E+11	8.30E+06	9.63E-01	Inf
CCJADE	1.80E-15	1.38E+02	3.74E+02	8.62E-01	4.85E-04	3.28E-13	0.00E+00
	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
MOS-SOCO2011	1.66E+05	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.69E+02	0.00E+00
jDElscop	3.17E+04	9.21E-08	0.00E+00	4.98E-08	0.00E+00	6.67E+02	4.03E-01
GaDE	1.55E+04	4.29E-04	3.36E-01	8.58E-04	2.90E-12	7.19E+02	7.72E-03
DECC-G	1.11E+05	1.78E+01	1.94E+02	1.76E+01	0.00E+00	3.86E+03	1.59E+02
CCJADE	1.92E+06	9.47E-01	0.00E+00	8.68E-01	4.46E+00	9.82E+01	3.46E-01
	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}		
MOS-SOCO2011	0.00E+00	0.00E+00	6.71E+01	0.00E+00	0.00E+00		
jDElscop	0.00E+00	0.00E+00	1.71E+02	3.28E-12	0.00E+00		
GaDE	8.40E-02	1.67E-12	2.18E+02	1.31E-07	2.10E-01		
DECC-G	1.84E+01	0.00E+00	1.98E+02	8.43E+00	1.12E+02		
CCJADE	0.00E+00	3.92E+00	7.83E+00	5.89E-01	0.00E+00		

Table 4: Average ranks of the optimization algorithms under comparison on the adopted test suite. The adjusted p-values p_{adj} were obtained adopting CCJADE as control method.

	MOS-SOCO2011	jDElscop	GaDE	DECC-G	CCJADE
Average rank	1.74	2.47	3.24	4.32	3.24
p_{adj}	0.014	0.178	1.000	0.070	-

optimizer already endowed with a high optimization ability, can easily lead to less remarkable improvements, or even to a worsening of the overall results. For this reason, we carried out a preliminary assessment of the search efficiency provided by the developed CCJADE algorithm. To this end, we selected from [32] the results provided by the three best-performing algorithms on the functions of Tables 1 and 2, namely MOS-SOCO2011 [31, 32], jDElscop [4] and GaDE [75], as well as by the CC-based optimizer DECC-G [74], which operates similarly to CCJADE.

To perform a fair comparison we adopted 5×10^6 fitness evaluations and 25 independent repetitions, as specified in [32]. Also, we used for CCJADE subcomponents of size 4 with 25 individuals and 6 JADE iterations per CC cycle. The mean results for the algorithms under comparison are reported in Table 3. We assessed the statistical differences between results through a rank-based approach following the guidelines suggested in [20, 12, 8] and using the *scmamp* [8], which is an R [50] package. To this purpose, for each function, the results obtained using each algorithm were converted to ranks by assigning rank 1 to the lowest mean error up to rank 5 to the highest. Then, the ranks obtained by each function were averaged. As can be seen from Table 4, the CCJADE algorithm performed equivalently to GaDE in terms of mean rank. Subsequently, we used the omnibus Friedman test [19] obtaining a p-value of 1.11E−05. The latter, with the adopted significance level of $\alpha = 0.05$ states the rejection of the null hypothesis of equality of all means. Therefore, adopting CCJADE as control algorithm, we applied pair-wise tests with a post-hoc correction for multiple comparison. To this purpose, we used the statistic from the Friedman test, based on the mutual difference between ranks, to compute the p-values corresponding to each comparison [12]. Then, due to the multiple pairwise comparisons, we used the Finner's [17] procedure of p-value adjustment as described in [20]. The algorithms for which the adjusted p-value was greater than the adopted level of significance were then considered as equivalent to CCJADE. According to the results in Table 4, the adopted CCJADE algorithm provided a lower optimization ability with respect to MOS-SOCO2011 and a performance statistically equivalent to the runner-up jDElscop. Therefore, overall, CCJADE provided a high optimization efficiency and proved suitable for the experimentation described in the following.

4.3. Size of subcomponents

To investigate the effect of the size of subcomponents, we used for d_k a set of values from 2 to 25. After some preliminary tests, and following the empirical study discussed in [66], for each value of d_k we selected a suitable size

Table 5: The used sizes of subcomponents and corresponding number of individuals.

d_k	2	4	8	20	25
N_{pop}	20	25	50	80	100

Table 6: Average rankings achieved for different sizes of subcomponents on the considered set of functions. For each type of metamodeling, statistically equivalent algorithms are highlighted.

d_k	CCJADE		SACCJADE-GP		SACCJADE-RBFN		SACCJADE-QPA		SACCJADE-SVR	
	Avg rank	p_{adj}	Avg rank	p_{adj}	Avg rank	p_{adj}	Avg rank	p_{adj}	Avg rank	p_{adj}
2	2.20	0.317	2.35	0.458	1.90	0.368	2.05	0.841	2.25	0.854
4	1.70	-	1.90	-	1.45	-	1.95	-	2.40	-
8	2.65	0.076	2.20	0.549	3.00	0.003	2.80	0.117	2.50	0.854
20	4.50	0.000	3.80	0.000	3.80	0.000	3.85	0.000	3.35	0.112
25	3.95	0.000	4.75	0.000	4.85	0.000	4.35	0.000	4.50	0.000

N_{pop} of population as shown in table 5. Then, for each value of d_k we carried out 50 independent optimizations of each test function, by adopting $N_{ite} = 6$ and using both the SACCJADE and the CCJADE.

The average achieved errors at the end of the $N_{FE} = 5 \times 10^5$ exact fitness evaluations are shown in Figure 2. Overall, the results show that the two highest values of d_k almost always determined a significant deterioration of the surrogate-assisted optimization. In some cases we observed a remarkable difference between the lower and the higher values of d_k (e.g., for functions f_1, f_5, f_6, f_7, f_{10}, f_{15} and f_{19}). This is likely to be related to the loss of accuracy in ranking correctly the population that affects the surrogate fitness as the dimension of the function domain increases. From this point of view, according to the graphs in Figure 2, the RBFN metamodel was characterized by the quickest decline of quality. In particular, we observed that even from $d_k = 8$, for some functions (e.g., f_1, f_9 f_{16}) the SACCJADE based on RBFN was unable to outperform CCJADE. Interestingly, the SVR-based approach proved the most scalable with respect to the size of decomposition (in 14 problems out of 19 it provided the best result with the larger size of decomposition). In particular, SVR led to the best results regardless of the size of decomposition for problem f_7.

Even considering the lowest values of d_k, for some test problems there were relevant differences between the average errors achieved using the different metamodeling. For example, in the case of f_5 with $d_k = 2$, the GP-assisted SACCJADE led to an average optimum of several order of magnitude lower than that provided by other values of d_k and metamodels. Also, for f_{19} the best average error of 1.62E−17 was achieved using the GP metamodel with $d_k = 8$. Instead, for the same optimization problem, the QPA-assisted SACCJADE provided its better average error of 2.75E−10 with subcomponents composed of $d_k = 2$ variables, while the RBFN-based approach could not outperform the CCJADE algorithm.

In general, the high variability of results was mainly due to the combined effect of: (*i*) the metamodel accuracy (higher with lower-sized subcomponents); (*ii*) the existence of an optimal value of d_k for the adopted CCJADE algorithm (see for example f_8, f_9, f_{11} and f_{18}); (*iii*) the different rate of consumption of the budget of exact fitness evaluations due to the different number of subcomponents. We carried out a statistical test to assess whether there was an overall superiority of a specific size of subcomponents, with regard to the adopted set of optimization problems. For each type of metamodeling, the comparison was conducted through the rank-based approach described above based on the Friedman's test. To this purpose, given a metamodeling strategy, for each function the results obtained using each of the five values of d_k were converted to ranks by assigning rank 1 to the lowest mean error up to rank 5 to the highest. For example, in the case of RBFN and function f_1 we assigned rank 1 to $d_k = 4$, rank 2 to $d_k = 2$, rank 3 to $d_k = 8$ and so on (see Figure 2). Then, the ranks obtained by each function were averaged. Subsequently, the omnibus Friedman test [19] led to a p-value much lower of the adopted significance level of $\alpha = 0.05$, stating the rejection of the null hypothesis of equality of all means. Then, to investigate which values of d_k were superior, we again applied the Finner's procedure of p-value adjustment for multiple comparisons [17, 20]. In particular, for each type of metamodeling, we assumed as control value of d_k that corresponding to the lowest rank. Each size d_k for which the adjusted p-value was greater than the adopted level of significance was then considered as equivalent to the best configuration, and the corresponding average rank was highlighted in Table 6.

In practice, the value $d_k = 4$ led to the best average ranking for all algorithms. However, for both GP and QPA a value of d_k between 2 and 8 provided statistically equivalent optimization performances. Instead, using the RBFN the best search efficiency was achieved, on average, with sizes of subcomponents of 2 and 4. It is noteworthy that the SVR-based

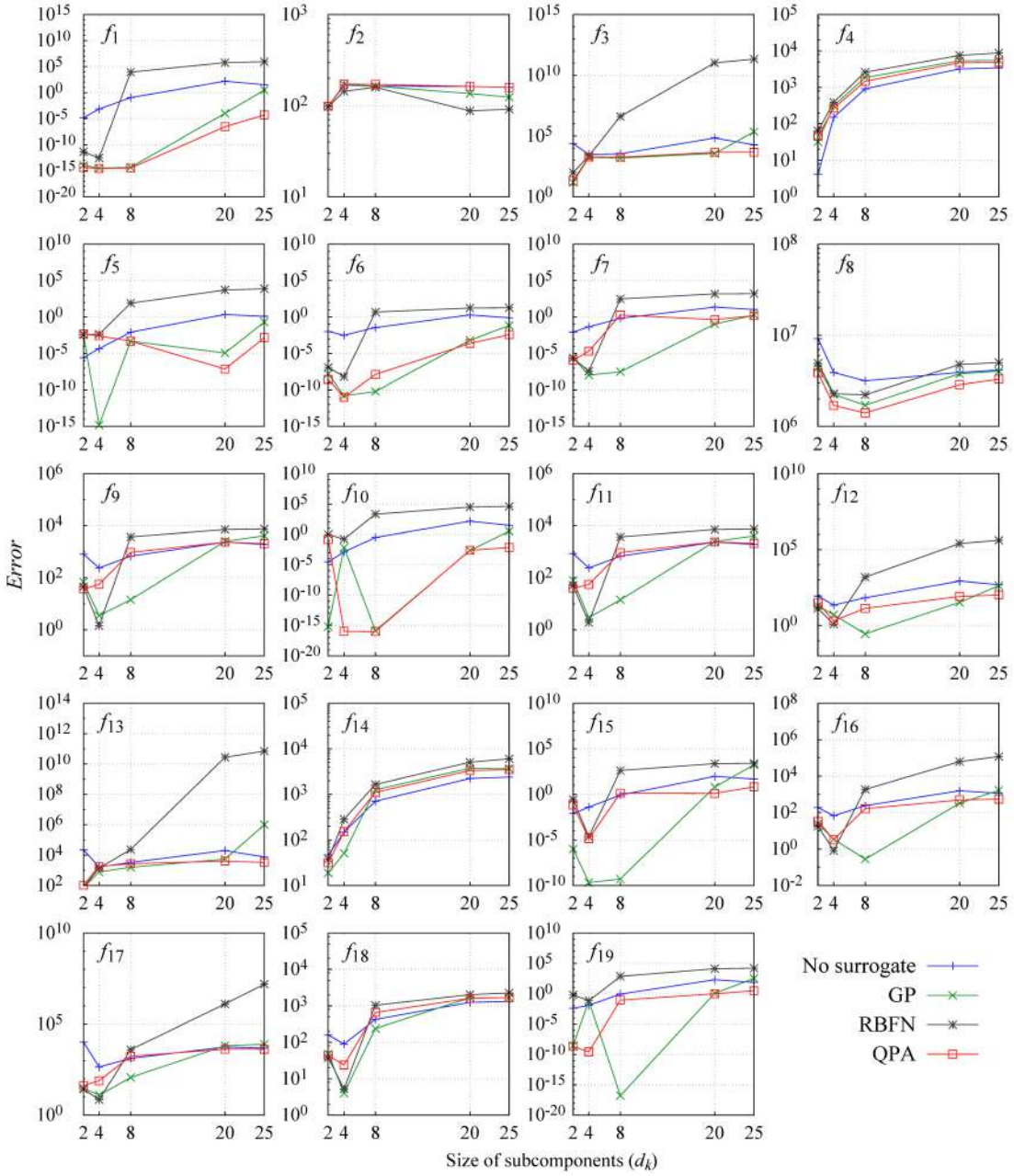

Figure 2: Effect of the size d_k of subcomponents on the average achieved error. The number of iterations per cycle was set to $N_{ite} = 6$.

CC optimizer could provide equivalent optimization efficiency up to the size of 20 variables per subcomponent. Clearly, such results can provide some useful indications in case of black-box functions, since, as shown in Figure 2 and mentioned above, a specific size of subcomponents could significantly improve the results in most cases.

4.4. Number of JADE iterations per CC cycle

In a CC optimizer, the number of iterations of the underlying optimizer performed at each cycle (i.e., N_{ite} in Algorithm 2) influences the frequency of RG as well as the frequency of information exchange between subcomponents. Low values

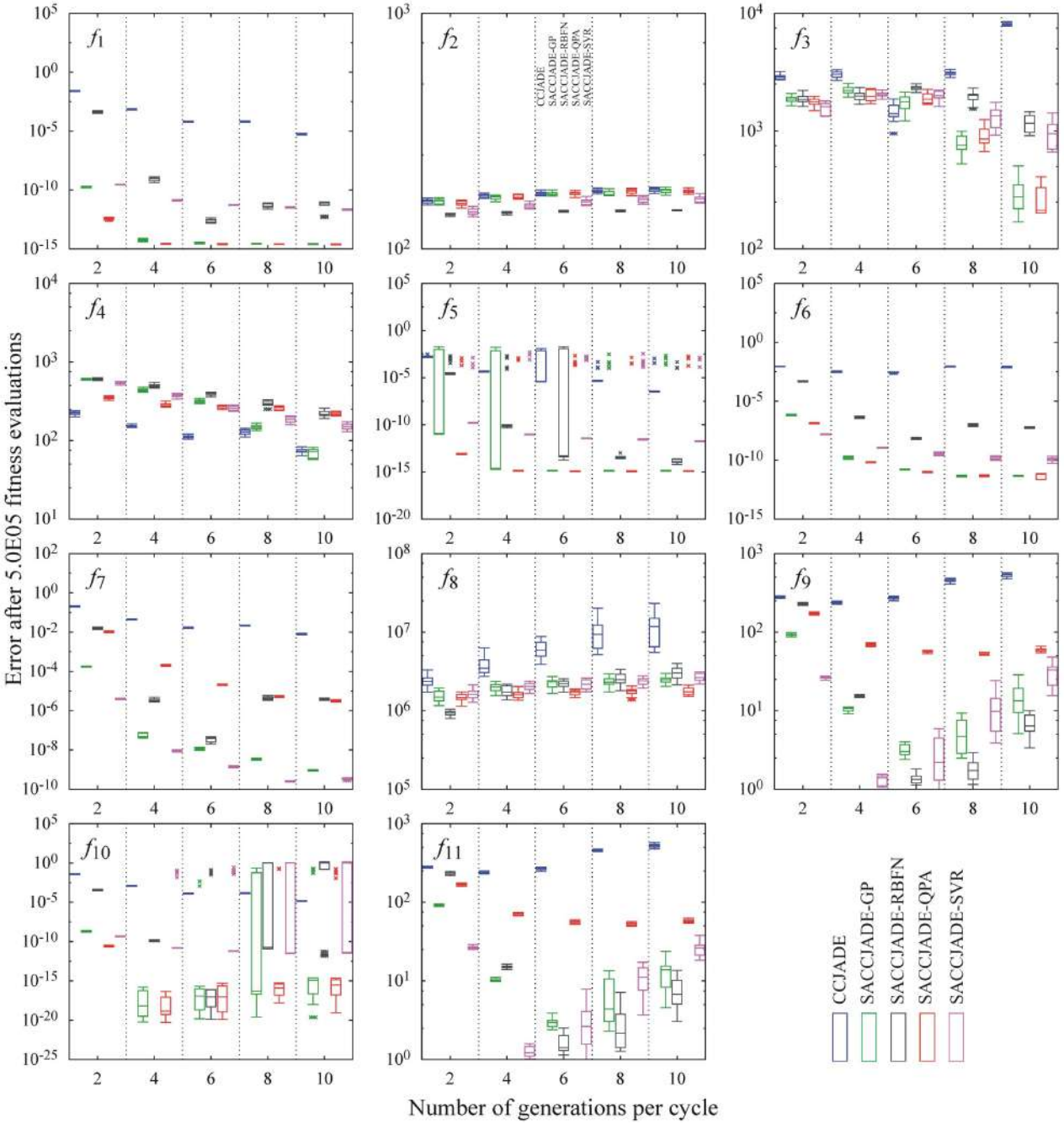

Figure 3: Box plots of results for functions f_1-f_{11} with different numbers N_{ite} of generations per cycle. The size of subcomponents was set to $d_k = 4$ with population size $N_{pop} = 25$.

of N_{ite} have been reported as beneficial in some cases of nonseparable functions [42]. In the proposed SACCJADE, the value of N_{ite} also determines: (*i*) the frequency with which the entire population is evaluated with the exact fitness; (*ii*) a restart from scratch of the accumulation of training patterns concerning the new subcomponents obtained after randomly regrouping the search directions. To investigate the combined effect of all the above factors, we carried out 50 independent optimizations of the test functions, for each value of N_{ite} in the set {2, 4, 6, 8, 10}. The optimizations were

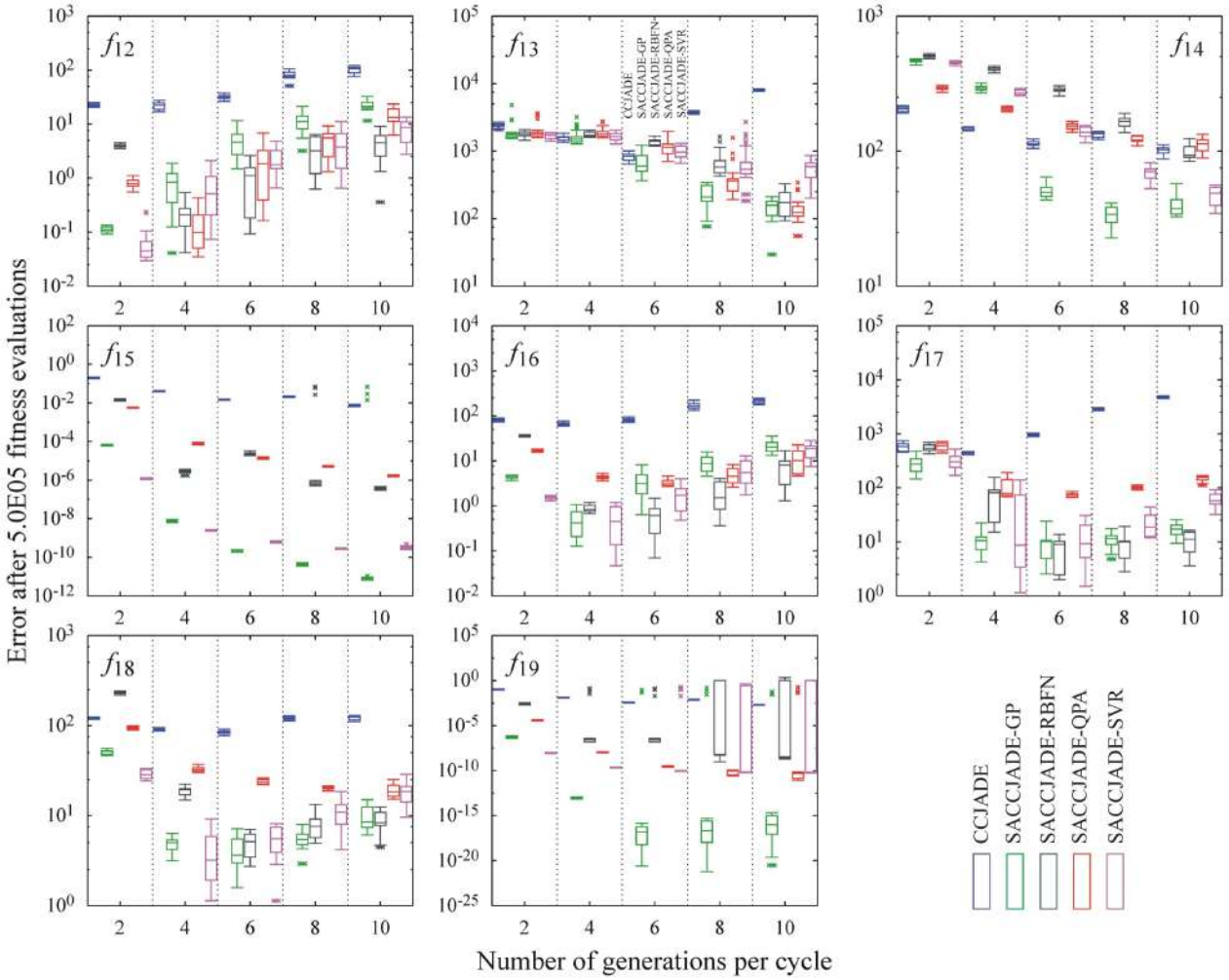

Figure 4: Box plots of results for functions f_{12}-f_{19} with different numbers N_{ite} of generations per cycle. The size of subcomponents was set to $d_k = 4$ with population size $N_{pop} = 25$.

executed using both CCJADE and its surrogate-assisted variations with subcomponents of size $d_k = 4$ and population of $N_{pop} = 25$ individuals, which proved reasonably effective for all algorithms according to Table 6.

The obtained results are summarized in Figures 3 and 4, where, for each function, algorithm and value of N_{ite}, we represent the box plots of the final errors achieved in the 50 independent runs. Note that the boxes are drawn around the regions between the first and third quartiles, with a horizontal line at the median value. Whiskers extend from the box to 1.5 times the inter-quartile range. Points that lie outside these limits are considered outliers and drawn individually. A first aspect to be noted is that, in most cases, the difference of achieved minima between CCJADE and SACCJADE (i.e., the gain of optimization accuracy provided by the metamodeling) is strongly influenced by the length N_{ite} of the CC cycle. For example, for the separable function f_6 a higher N_{ite} corresponds to a remarkably greater search efficiency for both the GP-assisted and the QPA-based SACCJADE. In contrast, in the case of the nonseparable function f_{12}, shorter CC cycles led to better optima, with the best median values provided by the GP-based SACCJADE with $N_{ite} = 2$ and by the QPA-assisted variant with $N_{ite} = 4$. Often, there was a clear optimal value of N_{ite}, among those investigated, as in the case of f_9 for which $N_{ite} = 6$ provided the lowest median error with the RBFN-based SACCJADE and f_{11}, for which the most effective algorith was SACCJADE-SVR with $N_{ite} = 4$.

Such different performances were also investigated in terms of statistical significance. More in detail, for each function we determined the best length of CC cycles providing the lowest average error. Then, using such an optimal value of N_{ite}

as control case, we used a series of pair-wise Mann-Whitney-Wilcoxon (MWW) tests with Holm correction [54], for the between-groups comparisons. The statistical tests, based on 5% significance level, showed that for each type of surrogate and test problem there is, in general, a range of significantly better values of the CC cycle length. In addition to Figures 3 and 4, which provide an overview of such an aspect, we show in Table 7 the details for functions f_6 and f_{10}. In the tables, the rows corresponding to the optimal values of N_{ite} are highlighted. Moreover, the last column shows the adjusted p-values computed according to the above-mentioned MWW statistical test. As can be seen, for function f_6 both the QPA-based and SVR-based SACCJADE could provide statistically equivalent results for CC cycles of lengths 8 and 10. For the remaining types of surrogate, a single value of N_{ite} proved optimal, namely $N_{ite} = 8$ using the GP and $N_{ite} = 6$ using the RBFN. As shown in Table 7, for function f_{10} there is always a single optimal value of N_{ite}. In particular, in this case all metamodels but SVR performed better with a CC length of 4 DE iterations.

In general, for the separable functions included in the test suite (i.e., f_1, f_4, f_6 and f_7) we observed better average results with a higher length of CC cycles. In particular, for f_4 and f_7 it seems that values of N_{ite} greater than 10 would have led to lower errors. However, the low number of separable functions included in the adopted test suite did not enable us to draw conclusions supported by a meaningful statistical analysis. Nevertheless, the fact that long CC cycles perform better for separable functions is not surprising since they can be optimized without dynamically regrouping and long cycles correspond to less resets of the archives used for training the surrogates. In contrast, for the nonseparable or partially separable problems considered in this study we could not find a general indication concerning the optimal duration of the CC cycle. More in detail, a Friedman's test [19] did not allow rejecting the null hypotheses of equivalence between the different values of N_{ite}. This is likely due to the many factors involved, as mentioned above, which influence the optimization with different intensities depending on the specific function.

Overall, the length of a CC cycle in the proposed SACCJADE can be very relevant and an optimal value is very problem-dependent. This suggests that it is meaningful to investigate strategies for adaptively finding the value of N_{ite}. Such an approach would be likely supported by the use of surrogates, which would allow a limited involvement of the original fitness function during the adaptive phase.

4.5. Comparison with CCJADE and MOS-SOCO2011

Focusing on a fixed configuration in terms of size of subcomponents, number of evolved individuals and length of CC cycles, we carried out a detailed comparison between CCJADE, MOS-SOCO2011 and the proposed SACCJADE with different types of metamodeling. In particular, given the results discussed above, we used $d_k = 4$ with $N_{pop} = 25$ and $N_{ite} = 6$ for SACCJADE. For the MOS-SOCO2011 algorithm, we used the settings described in [32]. However, for all algorithms we limited the maximum number of exact function evaluations to 5×10^5 instead of 5×10^6. Tables 8 and 9 show the statistics on the 50 independent runs. The corresponding median convergence plots are depicted in Figure 5.

For each function, we used as control algorithm the one that provided the lowest mean error. Then, we applied five pair-wise MWW tests, with Holm correction [54] and significance level of 5%, to determine the algorithms that performed equivalently to the best one. The corresponding adjusted p-values are reported in Tables 8 and 9, where the rows corresponding to the optimal algorithms are highlighted. As can be seen, in all cases except f_{10} and f_{17} there was a single best performing algorithm. More in detail, according to the statistics in Tables 8 and 9, in most cases a surrogate-assisted CC algorithm was able to significantly improve the results of the optimizer based only on the exact fitness function. In general, we observed that for problems that can be efficiently optimized using the adopted CCJADE algorithm, the use of surrogates within subcomponents led to a significant improvement of average results, as shown in Tables 8 and 9. This was the case, for example, of function f_1 (with all the metamodels), f_5 (with GP), f_{15} (with GP and SVR), f_6, f_{10} and f_{19} (with QPA), f_7 (with SVR), f_{11}, f_{12}, f_{16} and f_{17} (using the RBFN-based surrogate). For function f_8, which represents a rotated hyper-ellipsoid, the SACCJADE could only slightly outperform CCJADE. Interestingly, in the cases of the Bohachevsky's function f_{10} and of f_{19} (composed of 75% by f_{10}), for GP, RBFN and SVR, we noted a high difference between average and median value of the final error, due to the presence of some outliers (see also the box plot in Figure 4 for $N_{ite} = 6$).

In contrast, as can be seen from Tables 8 and 9, in most test problems that proved difficult for the CCJADE algorithm, the SACCJADE variants could not improve the achieved results. In particular, for f_2, f_3, f_4 and f_{13}, the surrogate-assisted approach did not provide practical advantages in terms of average results at the end of the given number of exact fitness evaluations. It is worth noting, however, that Figures 3 and 4 show that in the case of f_3 and f_{13} the results can be significantly improved by increasing the number of JADE iterations per CC cycle. Such a lack of improvements could be related to intrinsic inefficiencies of the adopted optimizer (i.e. JADE). Indeed, as reported in [32] there is a high variability

Table 7: Detail of the results for functions f_6 and f_10 with different values of N_{ite}. The size of subcomponents was set to $d_k = 4$ with $N_{pop} = 25$.

	N_{ite}	Median	Avg.	Std. Dev.	Best	Worst	p_{adj}
				f_6			
CCJADE	2	9.1E-003	9.0E-003	3.3E-004	8.1E-003	9.3E-003	0.001
	4	3.2E-003	3.2E-003	2.7E-004	2.7E-003	9.3E-003	0.000
	6	2.7E-003	2.6E-003	1.3E-004	2.4E-003	9.3E-003	-
	8	8.9E-003	8.8E-003	5.0E-004	2.4E-003	9.6E-003	0.000
	10	8.0E-003	8.1E-003	7.5E-004	2.4E-003	9.6E-003	0.000
SACCJADE-GP	2	7.2E-007	7.2E-007	5.4E-008	6.0E-007	8.0E-007	0.001
	4	1.7E-010	1.7E-010	3.5E-011	1.2E-010	2.2E-010	0.000
	6	1.6E-011	1.6E-011	7.9E-013	1.4E-011	1.7E-011	0.000
	8	4.3E-012	4.4E-012	4.0E-013	3.6E-012	5.3E-012	-
	10	4.7E-012	4.6E-012	2.4E-013	4.2E-012	5.0E-012	0.001
SACCJADE-RBFN	2	4.8E-004	4.8E-004	4.3E-005	4.1E-004	5.6E-004	0.001
	4	4.3E-007	4.2E-007	4.5E-008	3.3E-007	5.2E-007	0.000
	6	6.2E-009	6.5E-009	8.0E-010	5.6E-009	8.5E-009	-
	8	9.9E-008	1.0E-007	1.9E-008	7.2E-008	1.3E-007	0.000
	10	6.1E-008	6.1E-008	6.1E-009	5.1E-008	6.9E-008	0.000
SACCJADE-QPA	2	1.3E-007	1.3E-007	8.1E-009	1.2E-007	1.5E-007	0.001
	4	6.4E-011	6.6E-011	4.7E-012	5.8E-011	7.3E-011	0.000
	6	9.8E-012	9.7E-012	7.4E-013	8.5E-012	1.1E-011	0.000
	8	4.6E-012	4.7E-012	5.8E-013	3.8E-012	5.7E-012	0.500
	10	6.2E-012	4.6E-012	2.2E-012	2.2E-012	7.2E-012	-
SACCJADE-SVR	2	1.6E-008	1.6E-008	5.7E-010	1.5E-008	1.7E-008	0.001
	4	1.1E-009	1.1E-009	5.7E-011	1.0E-009	1.2E-009	0.000
	6	3.8E-010	3.6E-010	7.6E-011	2.3E-010	4.9E-010	0.000
	8	1.3E-010	1.4E-010	4.5E-011	9.4E-011	2.4E-010	0.077
	10	1.2E-010	1.2E-010	4.4E-011	5.1E-011	2.1E-010	-
				f_{10}			
CCJADE	2	3.7E-002	3.8E-002	3.4E-003	3.3E-002	4.4E-002	0.001
	4	1.3E-003	1.3E-003	8.5E-005	1.1E-003	4.4E-002	0.000
	6	1.3E-004	1.3E-004	1.1E-005	1.1E-004	4.4E-002	0.000
	8	1.5E-004	1.5E-004	1.2E-005	1.1E-004	4.4E-002	0.000
	10	1.4E-005	1.4E-005	8.8E-007	1.3E-005	4.4E-002	-
SACCJADE-GP	2	2.2E-009	2.2E-009	3.0E-010	1.7E-009	2.6E-009	0.001
	4	6.6E-019	3.1E-017	5.1E-017	6.3E-021	1.6E-016	-
	6	1.1E-017	2.3E-004	9.6E-004	1.5E-020	4.6E-003	0.033
	8	5.4E-017	2.9E-002	5.7E-002	2.6E-020	2.3E-001	0.000
	10	1.2E-015	2.9E-002	6.3E-002	2.4E-020	2.0E-001	0.000
SACCJADE-RBFN	2	3.7E-004	3.7E-004	4.3E-005	2.9E-004	4.5E-004	0.001
	4	1.2E-010	1.3E-010	2.0E-011	9.7E-011	1.7E-010	-
	6	9.3E-018	1.4E-002	3.1E-002	1.4E-020	1.3E-001	0.000
	8	1.9E-011	3.5E-001	4.9E-001	1.2E-011	1.0E+000	0.012
	10	1.0E+000	8.2E-001	4.8E-001	1.5E-012	1.5E+000	0.000
SACCJADE-QPA	2	2.6E-011	2.8E-011	4.3E-012	2.1E-011	3.6E-011	0.001
	4	1.5E-019	5.4E-018	1.2E-017	5.4E-021	4.7E-017	-
	6	9.2E-018	1.1E-016	1.7E-016	1.3E-020	4.9E-016	0.001
	8	1.2E-016	1.3E-002	4.8E-002	1.6E-018	2.0E-001	0.000
	10	3.1E-016	1.1E-002	3.6E-002	8.4E-020	1.8E-001	0.000
SACCJADE-SVR	2	4.7E-010	4.6E-010	3.5E-011	4.0E-010	5.3E-010	-
	4	1.6E-011	8.7E-003	2.5E-002	1.4E-011	1.1E-001	0.001
	6	6.2E-012	2.1E-002	5.2E-002	5.6E-012	2.8E-001	0.000
	8	3.3E-012	2.9E-001	4.6E-001	3.0E-012	1.0E+000	0.018
	10	4.2E-012	4.6E-001	5.4E-001	3.4E-012	1.4E+000	0.035

Table 8: statistics on the results achieved with the different algorithms. For each function, the algorithms with best average results, according to the MWW statistical test, are highlighted (p_{adj} indicates the adjusted p-value).

		CCJADE	MOS-SOCO2011	SACCJADE-GP	SACCJADE-RBFN	SACCJADE-QPA	SACCJADE-SVR
f_1	Median	6.0E-005	3.3E-005	2.9E-015	2.9E-013	2.5E-015	5.0E-012
	Avg.	6.1E-005	1.2E+001	3.0E-015	2.7E-013	2.5E-015	5.1E-012
	Std.Dev.	4.3E-006	1.5E+001	2.5E-016	7.7E-014	1.5E-016	2.6E-013
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	3.0E-004	-	2.0E-004
f_2	Median	1.7E+002	9.8E+001	1.7E+002	1.4E+002	1.7E+002	1.6E+002
	Avg.	1.7E+002	9.9E+001	1.7E+002	1.4E+002	1.7E+002	1.6E+002
	Std.Dev.	2.6E+000	3.6E+001	2.7E+000	7.9E-001	2.7E+000	4.2E+000
	p_{adj}	6.0E-004	-	5.0E-004	4.0E-004	3.0E-004	2.0E-004
f_3	Median	1.4E+003	3.3E+003	1.8E+003	2.3E+003	1.9E+003	2.0E+003
	Avg.	1.5E+003	3.2E+003	1.7E+003	2.3E+003	1.9E+003	2.0E+003
	Std.Dev.	2.3E+002	1.8E+002	2.4E+002	1.1E+002	1.8E+002	1.8E+002
	p_{adj}	-	6.0E-004	5.0E-004	4.0E-004	3.0E-004	2.0E-004
f_4	Median	1.1E+002	3.9E+001	3.2E+002	4.0E+002	2.7E+002	2.6E+002
	Avg.	1.1E+002	3.8E+001	3.2E+002	3.9E+002	2.7E+002	2.6E+002
	Std.Dev.	3.9E+000	7.1E+000	1.3E+001	1.3E+001	1.2E+001	1.5E+001
	p_{adj}	6.0E-004	-	5.0E-004	4.0E-004	3.0E-004	2.0E-004
f_5	Median	4.3E-006	3.2E-003	1.4E-015	5.4E-014	1.2E-015	3.8E-012
	Avg.	2.3E-003	1.5E-001	1.4E-015	4.1E-003	2.5E-004	1.5E-004
	Std.Dev.	4.0E-003	2.2E-001	6.2E-017	7.0E-003	8.6E-004	4.5E-004
	p_{adj}	6.0E-004	5.0E-004	-	4.0E-004	3.0E-004	2.0E-004
f_6	Median	2.7E-003	7.1E-003	1.6E-011	6.2E-009	9.8E-012	3.8E-010
	Avg.	2.6E-003	8.2E-003	1.6E-011	6.5E-009	9.7E-012	3.6E-010
	Std.Dev.	1.3E-004	8.1E-003	7.9E-013	8.0E-010	7.4E-013	7.6E-011
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	3.0E-004	-	2.0E-004
f_7	Median	1.7E-002	2.2E-003	1.2E-008	3.8E-008	2.1E-005	1.4E-009
	Avg.	1.7E-002	2.2E-003	1.1E-008	3.5E-008	2.1E-005	1.4E-009
	Std.Dev.	8.2E-004	1.4E-004	9.3E-010	8.4E-009	1.1E-006	9.6E-011
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	3.0E-004	2.0E-004	-
f_8	Median	5.9E+006	3.9E+006	2.3E+006	2.2E+006	1.7E+006	2.2E+006
	Avg.	6.3E+006	3.9E+006	2.2E+006	2.2E+006	1.7E+006	2.2E+006
	Std.Dev.	1.5E+006	1.7E+005	2.7E+005	2.2E+005	1.2E+005	2.8E+005
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	3.0E-004	-	2.0E-004
f_9	Median	2.7E+002	1.1E+002	3.0E+000	1.3E+000	5.6E+001	2.2E+000
	Avg.	2.7E+002	1.1E+002	3.1E+000	1.4E+000	5.6E+001	2.8E+000
	Std.Dev.	9.7E+000	2.4E+000	5.2E-001	1.8E-001	1.6E+000	1.7E+000
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	-	3.0E-004	2.0E-004
f_{10}	Median	1.3E-004	1.1E-005	1.1E-017	9.3E-018	9.2E-018	6.2E-012
	Avg.	1.3E-004	1.3E-002	3.4E-004	1.7E-002	1.1E-016	2.5E-002
	Std.Dev.	1.1E-005	3.5E-002	1.4E-003	4.2E-002	1.7E-016	6.4E-002
	p_{adj}	6.0E-004	5.0E-004	7.5E-001	5.0E-001	-	4.0E-004

of results for such test problems depending on the specific optimization algorithm. Moreover, functions f_3, f_4 and f_{13} are multimodal and it is known (e.g. [14, 46]) that, in these cases, the use of surrogate fitness can lead to an early stalling of the search process due to convergence at false global optima. In practice, in most cases of highly multimodal functions, a global model is not able to capture the details in the area close to the optima, even if it can reasonably describe with good approximation the general trend of the fitness landscape [14]. For example, as can be seen from the convergence plot in Figure 5, for function f_4, after an initial faster convergence, the SACCJADE algorithm became less efficient than the simpler CCJADE, with this latter showing a lower error at the end of the optimization. In these cases, an alternative approach to investigate is the use of surrogate-assisted LS as done in [45, 65].

Also compared with the MOS-SOCO2011 algorithm, the proposed approach was able to provide significantly better

Table 9: statistics on the results achieved with the different algorithms. For each function, the algorithms with best average results, according to the MWW statistical test, are highlighted (p_{adj} indicates the adjusted p-value).

		CCJADE	MOS-SOCO2011	SACCJADE-GP	SACCJADE-RBFN	SACCJADE-QPA	SACCJADE-SVR
f_{11}	Median	2.7E+002	1.1E+002	3.0E+000	1.4E+000	5.5E+001	2.7E+000
	Avg.	2.7E+002	1.1E+002	3.0E+000	1.6E+000	5.6E+001	3.0E+000
	Std.Dev.	9.7E+000	3.0E+000	4.0E-001	4.0E-001	2.3E+000	1.9E+000
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	-	3.0E-004	2.0E-004
f_{12}	Median	3.2E+001	4.3E+001	4.6E+000	1.1E+000	1.8E+000	1.8E+000
	Avg.	3.2E+001	5.3E+001	5.0E+000	9.8E-001	2.0E+000	2.3E+000
	Std.Dev.	2.8E+000	2.1E+001	2.5E+000	7.7E-001	1.8E+000	1.2E+000
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	-	7.0E-003	3.0E-004
f_{13}	Median	8.4E+002	2.6E+003	6.0E+002	1.4E+003	1.1E+003	9.7E+002
	Avg.	8.3E+002	2.9E+003	6.6E+002	1.4E+003	1.2E+003	9.8E+002
	Std.Dev.	9.9E+001	1.4E+003	2.0E+002	1.5E+002	4.1E+002	1.8E+002
	p_{adj}	6.0E-004	5.0E-004	-	4.0E-004	3.0E-004	2.0E-004
f_{14}	Median	1.1E+002	3.6E+001	5.0E+001	2.8E+002	1.5E+002	1.4E+002
	Avg.	1.1E+002	3.4E+001	5.1E+001	2.8E+002	1.5E+002	1.4E+002
	Std.Dev.	4.3E+000	4.3E+000	5.8E+000	1.3E+001	8.4E+000	1.1E+001
	p_{adj}	6.0E-004	-	5.0E-004	4.0E-004	3.0E-004	2.0E-004
f_{15}	Median	1.4E-002	1.6E-003	2.0E-010	2.2E-005	1.4E-005	6.3E-010
	Avg.	1.5E-002	6.4E-003	2.1E-010	2.3E-005	1.4E-005	6.3E-010
	Std.Dev.	5.8E-004	2.2E-002	2.3E-011	4.1E-006	1.1E-006	4.3E-011
	p_{adj}	6.0E-004	5.0E-004	-	4.0E-004	3.0E-004	2.0E-004
f_{16}	Median	8.0E+001	5.7E+001	3.1E+000	6.1E-001	3.2E+000	1.7E+000
	Avg.	8.1E+001	5.8E+001	3.5E+000	6.2E-001	3.3E+000	1.7E+000
	Std.Dev.	6.1E+000	5.2E+000	2.3E+000	3.9E-001	5.1E-001	9.7E-001
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	-	3.0E-004	2.0E-004
f_{17}	Median	9.7E+002	8.7E+002	1.0E+001	9.0E+000	7.4E+001	9.3E+000
	Avg.	9.7E+002	1.2E+003	8.5E+000	7.1E+000	7.5E+001	1.2E+001
	Std.Dev.	3.6E+001	1.7E+003	4.1E+000	4.0E+000	5.8E+000	8.8E+000
	p_{adj}	6.0E-004	5.0E-004	5.4E-002	-	4.0E-004	4.7E-001
f_{18}	Median	8.4E+001	2.6E+001	3.6E+000	5.1E+000	2.4E+001	5.5E+000
	Avg.	8.4E+001	2.8E+001	4.0E+000	5.0E+000	2.4E+001	5.4E+000
	Std.Dev.	3.7E+000	4.1E+000	1.5E+000	1.4E+000	1.4E+000	2.0E+000
	p_{adj}	6.0E-004	5.0E-004	-	9.7E-003	4.0E-004	1.4E-003
f_{19}	Median	3.5E-003	4.4E-004	1.5E-017	2.7E-007	2.7E-010	9.2E-011
	Avg.	3.7E-003	7.7E-003	4.5E-003	8.6E-003	2.7E-010	1.3E-002
	Std.Dev.	2.8E-004	2.8E-002	1.7E-002	3.7E-002	2.3E-011	3.7E-002
	p_{adj}	6.0E-004	5.0E-004	4.0E-004	3.0E-004	-	2.0E-004

results for most of the considered test problems. However, according to Tables 8 and 9, this was not the case of f_2, f_4 and f_{14} (composed of 75% by f_4). Also observing the convergence plots in Figure 5, this is likely due to the effectiveness of the LS phases used by the MOS-SOCO2011 and, again, suggests the importance of such a strategy for some problems.

According to the median convergence plots in Figure 5, which provide more insights on the optimization processes, the SACCJADE variants exhibited in most cases a superior efficiency over almost the entire budget of exact function evaluations. Frequently, the advantages provided by the proposed approach were very pronounced since the early stages of the optimization. For example, for functions f_1 and f_5 the most efficient variant of SACCJADE (based on QPA and SVR) required only $1.0E05$ exact fitness evaluations to provide an optimum value below $1.0E{-}05$. It is interesting to note that the SVR-based variant showed very often the quickest convergence, such as in the cases of f_6, f_7, f_{10} and f_{19}. It is also worth noting that, according to the convergence plots, for several functions (e.g. f_6, f_7, f_8, f_9, f_{11}, f_{15}) the SACCJADE variants provided a constantly superior speed of convergence and that a greater number of exact fitness evaluations would likely have provided a greater superiority of results, compared to CCJADE. Interestingly, for some of the test problems in which SACCJADE proved not effective (e.g., f_3 and f_{13}), we observed an initial advantage of all the surrogate-assisted

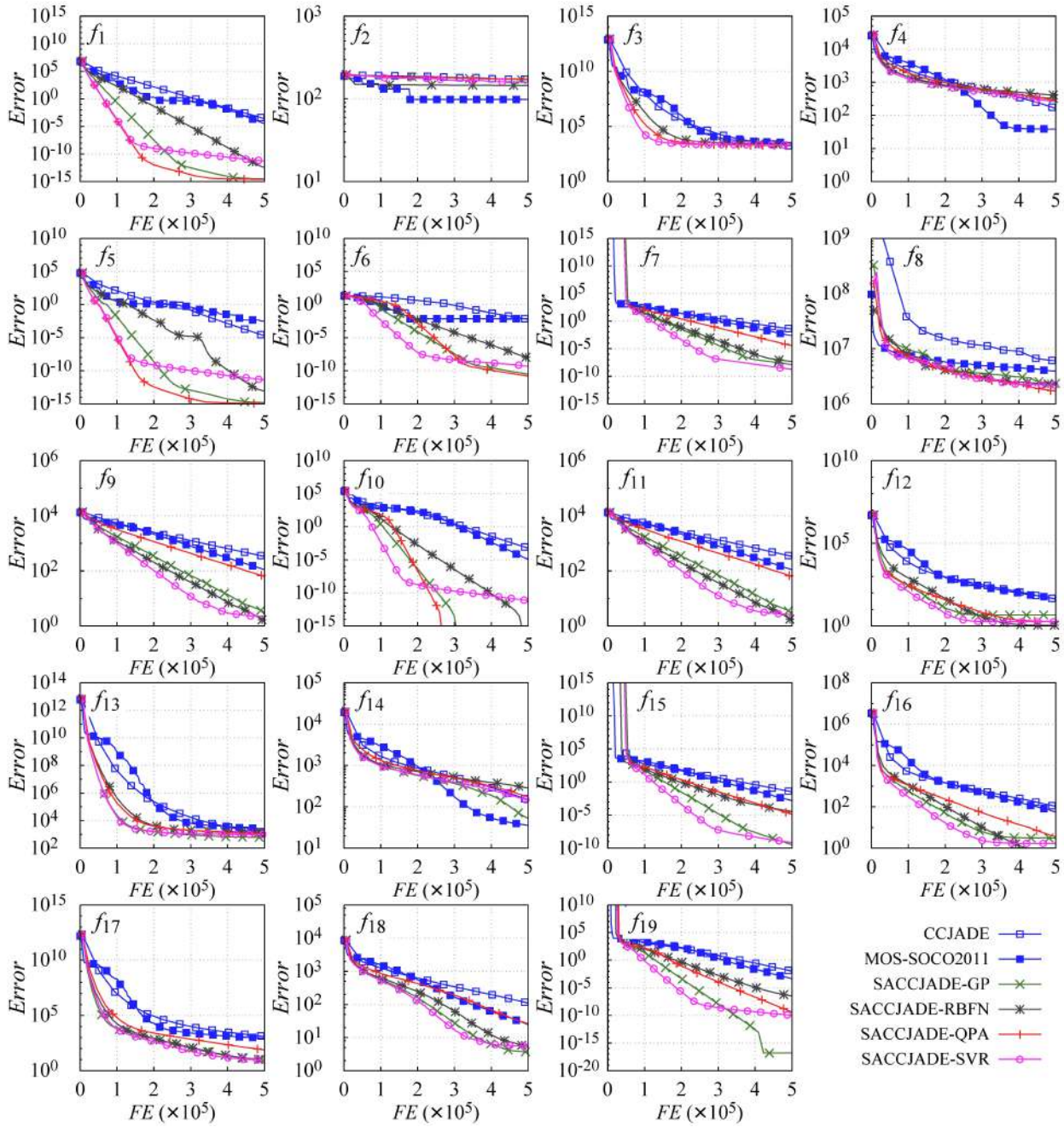

Figure 5: Median error as a function of the number of exact fitness evaluations for CCJADE, MOS-SOCO2011 and for SACCJADE, with the different metamodeling techniques.

variants, followed, however, by a convergence stalling in the final part of the process, due to the reasons discussed above.

4.6. Comparison between different types of metamodeling

In Figure 6 we show the average percentage gain Δ_{FE} of exact fitness evaluations provided by the different variants of our SACCJADE algorithm. The values Δ_{FE} are computed as $100 \cdot (N_{FE} - N^*_{FE})/N_{FE}$, where N_{FE} is the adopted budget of exact fitness evaluations (see Algorithm 1) and N^*_{FE} is the number of exact fitness evaluations needed to the SACCJADE

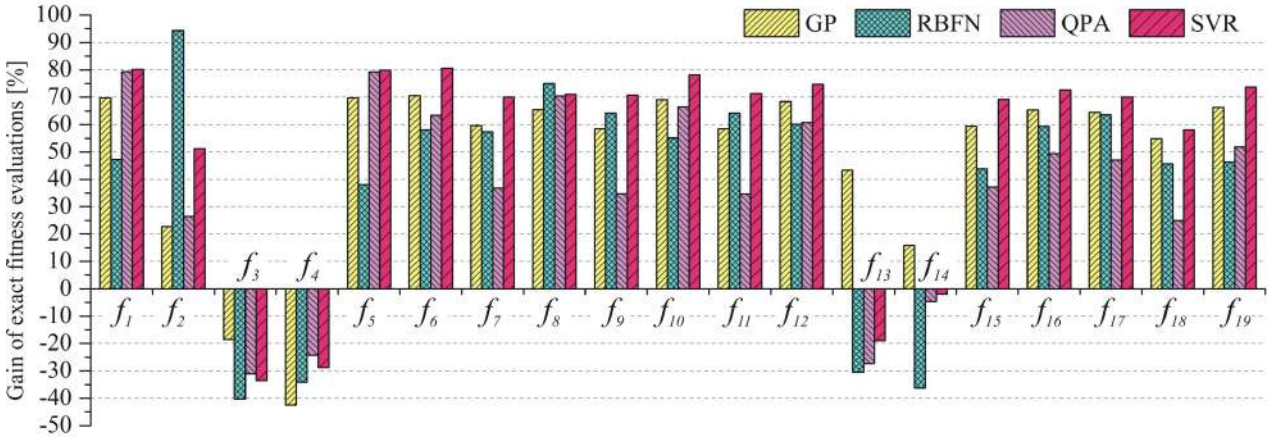

Figure 6: Median gains of exact fitness evaluations provided by the SACCJADE algorithms over CCJADE.

algorithm for achieving the same result provided by CCJADE. For example, using GP for problem f_6 the gain was 70%, which means that SACCJADE-GP provided the final result of $2.6E{-}03$ attained by CCJADE with a gain of 350000 exact fitness evaluations. Using SVR for the same problem, the gain was even greater. Overall, by considering the cases in which SACCJADE was successful in the adopted test cases, we obtained an average gain of original fitness functions of about 58% with GP and RBFN, and 71% with SVR. Clearly, in Figure 6 the meaningfulness of the gains of fitness evaluations shown is greater when the CCJADE algorithm proved effective in the optimization task. That said, it can be seen that for most optimization problems the computational gain, in terms of exact fitness evaluations, is above 50% and the achieved error level is useful for most practical applications (e.g., f_1, f_5, f_6, f_7, f_{10}, f_{15} and f_{19}).

We also assessed whether one of the used metamodeling techniques was significantly better, with regard to the average performance in terms of gain of exact fitness evaluations. To this purpose, we transformed in average ranks the data of Figure 6, obtaining 2.37, 3.16, 3.11 and 1.37 for the GP, RBFN, QPA and SVR respectively. Then, the best mean rank was provided by the SVR-based SACCJADE algorithm. The omnibus Friedman test led to a p-value of $2.614e-05$, suggesting further pair-wise comparisons. Then, by adopting the SVR-based algorithm as control method, we performed three Finner's post-hoc comparisons, which provided adjusted p-values of 0.02 for GP and below 0.01 for RBFN and QPA, all corresponding to the rejection of equivalence hypothesis for $\alpha = 0.05$. Therefore, in terms of gain of exact fitness evaluations, the SVR metamodeling technique performed better than the remaining approaches.

It is worth noting that the assessed average superiority of SVR in terms of gain of exact fitness evaluations is due to the fast speed of convergence, which is often followed by a stagnation. In fact, Tables 8 and 9 show that GP and RBFN can more often achieve better results in the long run. Albeit to a less extent, even QPA can provide some remarkable results. For example, it proved the most effective for functions f_1, f_6, f_{10} and f_{19}. This is particularly relevant considering that, as discussed later, the computational costs of GP can be significantly greater than simpler metamodeling approaches like QPA. In general, the success of a simple local approximation like QPA depends on the fitness landscape generated by the objective function. In fact, for both unimodal problems generated by the sphere function f_1 and Bohachevsky function f_{10} (which is also the main component of f_{19}), the trend in the neighborhood of a point can be effectively modeled as a quadratic polynomial. However, also for the multimodal Ackley function f_6 the QPA was able to generate an accurate local approximation. Instead, the cheaper local metamodeling was clearly inferior for fitness landscapes characterized by singularities, like the Schwefel's Problem 2.22 (f_7) or the Shaffer function (f_9 and f_{11}).

4.7. Advantages provided by SACCJADE in terms of computing time

The computing time for training and using the GP metamodel is mainly due to the training process, consisting in the log-likelihood maximization, which requires several inversions of the $n_r \times n_r$ matrix collecting the covariances between the n_r training points. Therefore, the asymptotic time complexity of such phase is $O(N_{opt}\, n_r^3)$, where N_{opt} is the number of iterations performed by the log-likelihood maximization algorithm. For this reason, the GP-based modeling tends to become very expensive as the amount of data used for training increases. However, given that some quantities can be

Table 10: Total CPU time T_S (in seconds) taken by the surrogate training and evaluation for the whole optimization in the case of function f_{10} and for different sizes of subcomponents. In this experiment we used $N_{ite} = 6$.

	$d_k = 2$, $N_{pop} = 20$				$d_k = 4$, $N_{pop} = 25$				$d_k = 8$, $N_{pop} = 50$			
	GP	RBFN	QPA	SVR	GP	RBFN	QPA	SVR	GP	RBFN	QPA	SVR
T_S	351.94	34.57	3.66	157.83	569.96	68.78	13.67	131.91	2092.44	190.96	295.57	276.49
Model training	343.41	33.64	3.64	156.39	556.94	67.15	13.63	130.48	2053.51	187.35	295.36	271.37
Surrogate evaluation	4.82	0.93	0.02	1.44	7.41	1.63	0.04	1.43	21.92	3.61	0.21	5.12
Variance evaluation	3.71	-	-	-	5.61	-	-	-	17.01	-	-	-

cached after the first use, the first evaluation of an individual costs $O(n_r^3)$ and the remaining only $O(n_r)$. As for the variance estimation, supposing that the Cholesky factorization of the covariance matrix has been cached, it corresponds to the cost of the forward and backward substitution, that is $O(n_r^2)$.

The cost of training the RBFN metamodel through the gradient descent approach is $O(N_{opt}\, d_k\, n_r\, n_c)$, where N_{opt} is the number of iterations. Since we set n_r to the size of the training archive and n_c at a fixed fraction of n_r, we have that the asymptotic cost of RBFN training is $O(N_{opt}\, d_k\, |\mathcal{A}|^2)$. Therefore, the computing time required by RBFN-based surrogates is less sensitive to the amount of used data, when compared with the GP counterpart. Moreover, according to Equation (3), the cost of each RBFN-based evaluation is $O(d_k\, n_c)$.

The main computational cost of the adopted QPA metamodel is due to the inversion of an $n_p \times n_p$ matrix, which is $O(n_p^3)$. However, since n_p is quadratic on d_k, the actual complexity of evaluating each individual with the QPA model is $O(d_k^6)$, which can become not affordable even for moderately high values of d_k.

In the case of SVR the training time complexity is between $O(d_k\, |\mathcal{A}|)$ and $O(d_k\, |\mathcal{A}|^2)$, depending on the number of SMO iterations actually performed. The time cost of each SVR-based evaluation is $O(d_k\, |\mathcal{A}|)$.

Overall, the computational costs for building GP, RBFN and SVR are sensitive to the size of the used archive of past exact fitness evaluations, while the QPA cost mainly depends on the size of subcomponents. However, the size of the used archive is, to some extent, related to the dimensionality of subcomponents through the number of used individuals.

That said, the actual time cost related to metamodeling in SACCJADE depends on several factors. In fact, the final number of surrogate trainings and evaluations is determined by the number and size of subcomponents, the number of evolved individuals and also by the accuracy of the surrogate itself. For example, due to the exact re-evaluation of some individuals at each JADE generation, a higher number of subcomponents tends to require less CC cycles to consume the available budget of exact fitness evaluations. Also, a surrogate of higher quality requires less re-evaluations to ensure the condition of having the best individual with exact fitness (see Algorithm 2, lines 21-26). On the other hand, a lower number of CC cycles implies less metamodel training and evaluations at the end of the process. To give an idea of the additional computing effort required by metamodeling in the proposed algorithm, we show in Table 10 the CPU time T_s taken by the surrogate training and evaluation for the whole optimization in the case of function f_{10}, for different sizes of subcomponents and number of individuals. To perform the computation we used the setting specified in section 4.1, with $N_{ite} = 6$. The elapsed times were collected on a PC with an Intel Core i7-4600U CPU. As can be seen, the cost of GP increases rapidly from the small subcomponents with 20 individuals to those based on bigger populations. Also, the overall QPA training is very small for $d_k = 2$ and significantly higher for $d_k = 8$. Table 10 also shows that the total cost of metamodeling is always reasonable, even if a number of surrogates are built during the optimization process.

Clearly, the advantages in terms of computing times that the method can bring to concrete applications depend on both the cost of the objective function and the gain of exact fitness evaluations. More in detail, given the time t_f required by each evaluation of the original objective function, a percentage gain Δ_{FE} in terms of number of fitness evaluations corresponds to a time saving of $T_{FE} = \Delta_{FE}\, N_{FE}\, t_f/100$. Therefore, the surrogate-assisted approach is suitable when the additional total time T_S required by metamodeling is lower than T_{FE}.

To quantify in a concrete case the benefits provided by the surrogate-assisted CC we considered the statistics achieved for function f_{10} for $d_k = 4$ and $N_{pop} = 25$. According to Figure 6, using SACCJADE-GP led to a gain of 345000 exact fitness evaluations with an extra cost of $T_S = 569.96s$ due to the surrogate training and evaluation. Therefore, for an hypothetical f_{10} with computing times greater than $T_0^{GP} = T_S/345000 = 1.65E{-}03s$, the GP-assisted approach would provide a computational gain. However, in the case of f_{10} also the QPA-based approach was very effective, by providing a gain of 332500 original fitness evaluations at the extra cost of only $13.67s$. Therefore, SACCJADE-QPA would be

Table 11: Total time required by CCJADE and SACCJADE-GP for fitness evaluations to achieve the same result. The comparisons were computed for different hypothetical time costs of the objective function f_{10} with $d_k = 4$ and $N_{pop} = 25$ (see Table 10). The value $T_0 = 1.65E\text{-}03s$ corresponds to a balance between the gain of original objective function evaluations and the overhead due to surrogates.

	Time cost of objective function				
	T_0	$2 \times T_0$	$5 \times T_0$	$10 \times T_0$	$100 \times T_0$
Total time for fitness evaluation with CCJADE [h]	0.23	0.46	1.15	2.29	22.95
Total time for fitness evaluation with SACCJADE-GP (original plus surrogate) [h]	0.23	0.30	0.51	0.87	7.27
Time gain [h]	0.00	0.16	0.63	1.42	15.67
Time gain [%]	0%	35%	55%	62%	68%

suitable for an objective function f_{10} with computing times greater than $T_0^{QPA} = 4.11E{-}05s$. In Table 11 we show the overall computing times required by CCJADE and SACCJADE-GP for fitness evaluations (i.e. original plus surrogate) to achieve the same result with different hypothetical costs of the objective function. As can be seen, even an objective function requiring $0.17s$ corresponds to about $23h$ of CPU time during an optimization based on $5{\times}10^5$ fitness evaluations. Instead, using SACCJADE-GP the same result can be achieved in about $7.3h$, with a gain of 68%. Therefore, it is clear that SACCJADE can represent a valuable approach for a wide range of applications involving complex objective functions defined in high-dimensional spaces.

5. Conclusions and future work

According to the presented computational study, the use of approximate fitness within the subcomponents of a CC optimization algorithm proved effective in boosting the search efficiency in most of the considered high-dimensional optimization problems. Moreover, we showed that the additional computational cost for surrogate training and evaluation can be considered acceptable even for relatively inexpensive objective functions.

We investigated the main parameters potentially affecting the behaviour of the used algorithms, such as the size of decomposition and the length of CC cycles. In most problems, the SACC method could provide a significant acceleration of convergence using low-sized subcomponents (i.e., composed of 2 or 4 variables). However, the optimal length of a CC cycle, which also determines the frequency with which the entire population is evaluated with the original objective function, proved quite problem-dependent. Therefore, future work might include the development of an adaptive SACC algorithm in which the number of iterations per CC cycle performed is automatically determined.

Furthermore, we compared four typical metamodeling techniques, namely GP, RBFN, SVR, used as global surrogates, and QPA as local fitness approximation. Overall, the best performing surrogates were those based on GP and RBFN. However, the SVR-based approach led to a faster convergence in most cases, which imply a higher gain of exact fitness evaluations when the computational budged is limited. Nevertheless, we showed that, even using the most time-consuming GP-based approach, the surrogate-assisted CC optimizer could provide valuable gains of computing time for objective functions with relatively low computational cost. Future work could consider adopting suitable strategies for automatically choosing the most convenient metamodeling approach in case of black-box functions [1].

In some of the considered test problems, SACC could not improve the results of the baseline CC optimizer. This was especially the case of some multimodal problems, for which it is known that the use of approximate fitness can lead to an early stalling of the search process [14, 46]. According to the literature, a more effective approach to consider in future work is the use of surrogate-assisted LS [45] within the subcomponents.

In our approach, we ensure that the best individual according to the surrogate is re-evaluated with the exact fitness. As a result, the training sets are enriched over the iterations of each CC cycle and this allows to slowly update the fitness models during the process. However, less straightforward infill sampling criteria to update the surrogate model should be investigated in the future [28, 18].

In the presented empirical investigation, we did not consider decompositions algorithms to be applied before the optimization in order to create subcomponents in which all the interacting variables are grouped together [44]. On the one hand, such a static decomposition algorithms typically require a high number of objective function evaluations (i.e., $O(d^2)$ in [44]). On the other hand, avoiding the need of frequent random re-groupings during optimization would enable longer CC cycles, likely leading to a greater efficiency of the surrogate-assisted CC approach. Therefore, in future work it is worth investigating the advantages of using the SACC approach to tackle partially separable problems.

Another line of research that deserves to be studied more extensively was introduced in [37], where metamodeling was exploited in the decomposition phase but not during optimization. It should be investigated whether it is possible to exploit the approximation of objective function computed during the surrogate-assisted decomposition to improve efficiency in building metamodels within the subcomponents.

Overall, there are several promising research directions that are worth exploring to further advance efficiency of the proposed SACC optimizer, when applied to LSGO problems characterized by complex objective functions.

Acknowledgements

This study was supported by the research grants for the project "Large Scale Optimization of Computationally Expensive Objective Functions" funded by Fondazione di Sardegna (2015).

References

References

[1] S. Bagheri, W. Konen, T. Bck, Online selection of surrogate models for constrained black-box optimization, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1–8.

[2] I. Blecic, A. Cecchini, G. A. Trunfio, How much past to see the future: a computational study in calibrating urban cellular automata, International Journal of Geographical Information Science 29 (3) (2015) 349–374.

[3] M. Blum, M. A. Riedmiller, Optimization of Gaussian process hyperparameters using Rprop, in: 21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013, 2013.

[4] J. Brest, M. S. Maucec, Self-adaptive differential evolution algorithm using population size reduction and three strategies, Soft Comput. 15 (11) (2011) 2157–2174.

[5] D. Bueche, N. Schraudolph, P. Koumoutsakos, Accelerating evolutionary algorithms with Gaussian process fitness function models, IEEE Trans. on Systems, Man, and Cybernetics: Part C 35 (2) (2004) 183 – 194.

[6] M. D. Buhmann, Radial Basis Functions: Theory and Implementations, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2003.

[7] R. H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM Journal on Scientific Computing 16 (5) (1995) 1190–1208.

[8] B. Calvo, G. Santafé Rodrigo, scmamp: statistical comparison of multiple algorithms in multiple problems, The R Journal, Vol. 8/1, Aug. 2016.

[9] T. Chai, Y. Jin, B. Sendhoff, Evolutionary complex engineering optimization: opportunities and challenges, IEEE Computational Intelligence Magazine 8 (3) (2013) 12–15.

[10] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, in: Parallel Problem Solving from Nature, PPSN XI, vol. 6239 of LNCS, Springer, 2010, pp. 300–309.

[11] S. Cheng, Y. Shi, Q. Qin, R. Bai, Swarm intelligence in big data analytics, in: H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li, X. Yao (eds.), Intelligent Data Engineering and Automated Learning - IDEAL 2013, vol. 8206 of LNCS, Springer, 2013, pp. 417–426.

[12] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 3–18.

[13] M. El-Beltagy, A. Keane, Evolutionary optimization for computationally expensive problems using Gaussian processes, in: Proceedings of International Conference on Artificial Intelligence, CSREA, 2001, pp. 708–714.

[14] M. A. El-Beltagy, P. B. Nair, A. J. Keane, Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations, in: Proceedings of GECCO - Volume 1, GECCO'99, 1999, pp. 196–203.

[15] H. Ergun, D. Van Hertem, R. Belmans, Transmission system topology optimization for large-scale offshore wind integration, Sustainable Energy, IEEE Transactions on 3 (4) (2012) 908–917.

[16] A. A. Esmin, R. Coelho, S. Matwin, A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data, Artificial Intelligence Review (2013) 1–23.

[17] H. Finner, On a monotonicity problem in step-down multiple test procedures, Journal of the American Statistical Association 88 (423) (1993) 920–923.

[18] A. I. Forrester, A. J. Keane, Recent advances in surrogate-based optimization, Progress in Aerospace Sciences 45 (1) (2009) 50 – 79.

[19] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701.

[20] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.

[21] C. Garca-Martnez, M. Lozano, F. Herrera, D. Molina, A. Snchez, Global and local real-coded genetic algorithms based on parent-centric crossover operators, European Journal of Operational Research 185 (3) (2008) 1088 – 1113.

[22] A. Giunta, L. Watson, A comparison of approximation modeling techniques: Polynomial versus interpolating models, Tech. Rep. 98-4758, AIAA (1998).

[23] C. Goh, D. Lim, L. Ma, Y. Ong, P. Dutta, A surrogate-assisted memetic co-evolutionary algorithm for expensive constrained optimization problems, in: Evolutionary Computation (CEC), 2011 IEEE Congress on, 2011, pp. 744–749.

[24] A. Hameed, D. Corne, D. Morgan, A. Waldock, Large-scale optimization: Are co-operative co-evolution and fitness inheritance additive?, in: 2013 13th UK Workshop on Computational Intelligence (UKCI), 2013, pp. 104–111.
[25] A. Hameed, A. Kononova, D. Corne, Engineering fitness inheritance and co-operative evolution into state-of-the-art optimizers, in: 2015 IEEE Symposium Series on Computational Intelligence, 2015, pp. 1695–1702.
[26] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Comput. 9 (1) (2005) 3–12.
[27] Y. Jin, Surrogate-assisted evolutionary computation: Recent advances and future challenges, Swarm and Evolutionary Computation 1 (2) (2011) 61 – 70.
[28] D. R. Jones, A taxonomy of global optimization methods based on response surfaces, Journal of Global Optimization 21 (4) (2001) 345–383.
[29] S. Kimura, K. Ide, A. Kashihara, M. Kano, M. Hatakeyama, R. Masui, N. Nakagawa, S. Yokoyama, S. Kuramitsu, A. Konagaya, Inference of s-system models of genetic networks using a cooperative coevolutionary algorithm, Bioinformatics 21 (7) (2004) 1154.
[30] D. E. King, Dlib-ml: A machine learning toolkit, Journal of Machine Learning Research 10 (2009) 1755–1758.
[31] A. LaTorre, S. Muelas, J. M. Peña, A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test, Soft Comput. 15 (11) (2011) 2187–2199.
[32] A. LaTorre, S. Muelas, J. M. Peña, A comprehensive comparison of large scale global optimizers, Inform. Sci. 316 (2015) 517–549.
[33] D. Lim, Y. Jin, Y.-S. Ong, B. Sendhoff, Generalizing surrogate-assisted evolutionary computation, IEEE Trans. Evol. Comp. 14 (3) (2010) 329–355.
[34] B. Liu, Q. Zhang, G. G. E. Gielen, A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems, IEEE Transactions on Evolutionary Computation 18 (2) (2014) 180–192.
[35] M. Lozano, D. Molina, F. Herrera, Editorial: scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, Soft Comput. 15 (11) (2011) 2085–2087.
[36] Y. Lu, S. Wang, S. Li, C. Zhou, Particle swarm optimizer for variable weighting in clustering high-dimensional data, Machine Learning 82 (1) (2011) 43–70.
[37] S. Mahdavi, M. E. Shiri, S. Rahnamayan, Cooperative co-evolution with a new decomposition method for large-scale optimization, in: 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 1285–1292.
[38] S. Mahdavi, M. E. Shiri, S. Rahnamayan, Metaheuristics in large-scale global continues optimization: A survey, Inform. Sci. 295 (0) (2015) 407 – 428.
[39] Y. Mei, M. N. Omidvar, X. Li, X. Yao, A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization, ACM Trans. Math. Softw. 42 (2) (2016) 13:1–13:24.
[40] P. B. Nair, A. J. Keane, Passive vibration suppression of flexible space structures via optimal geometric redesign, AIAA journal 39 (7) (2001) 1338–1346.
[41] M. N. Omidvar, X. Li, K. Tang, Designing benchmark problems for large-scale continuous optimization, Information Sciences 316 (2015) 419 – 436.
[42] M. N. Omidvar, X. Li, Z. Yang, X. Yao, Cooperative co-evolution for large scale optimization through more frequent random grouping, in: IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.
[43] M. N. Omidvar, Y. Mei, X. Li, Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms., in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2014, pp. 1305 – 1312.
[44] M. N. Omidvar, M. Yang, Y. Mei, X. Li, X. Yao, DG2: A faster and more accurate differential grouping for large-scale black-box optimization, IEEE Transactions on Evolutionary Computation PP (99) (2017) 1–1.
[45] Y. Ong, A. Keane, P. Nair, Surrogate-assisted coevolutionary search, in: Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02, vol. 3, 2002, pp. 1140–1145.
[46] Y. Ong, P. Nair, A. Keane, Evolutionary optimization of computationally expensive problems via surrogate modeling, AIAA Journal 41 (4) (2003) 687–696.
[47] Y.-S. Ong, Z. Zhou, D. Lim, Curse and blessing of uncertainty in evolutionary algorithm using approximation, in: 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 2928–2935.
[48] J. C. Platt, Advances in kernel methods, chap. Fast Training of Support Vector Machines Using Sequential Minimal Optimization, MIT Press, Cambridge, MA, USA, 1999, pp. 185–208.
[49] M. A. Potter, K. A. De Jong, A cooperative coevolutionary approach to function optimization, in: Parallel Problem Solving from Nature - PPSN III, vol. 866 of LNCS, Springer-Verlag, 1994, pp. 249–257.
[50] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2013). URL `http://www.R-project.org/`
[51] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.
[52] R. Regis, C. Shoemaker, Constrained global optimization of expensive black box functions using radial basis functions, Journal of Global Optimization 31 (1) (2005) 153–171.
[53] F. Schwenker, H. A. Kestler, G. Palm, Three learning phases for radial-basis-function networks, Neural Networks 14 (4-5) (2001) 439 – 458.
[54] D. J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 4th ed., Chapman & Hall/CRC, 2007.
[55] R. E. Smith, B. A. Dike, S. A. Stegmann, Fitness inheritance in genetic algorithms, in: Proceedings of the 1995 ACM Symposium on Applied Computing, SAC '95, ACM, New York, NY, USA, 1995, pp. 345–350.
[56] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (3) (2004) 199–222.
[57] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[58] C. Sun, J. Ding, J. Zeng, Y. Jin, A fitness approximation assisted competitive swarm optimizer for large scale expensive optimization problems, Memetic Computing.
[59] J. Sun, J. M. Garibaldi, C. Hodgman, Parameter estimation using metaheuristics in systems biology: a comprehensive review, IEEE/ACM Transactions on Computational Biology and Bioinformatics 9 (1) (2012) 185–202.
[60] K. Tang, X. Li, P. N. Suganthan, Z. Yang, T. Weise, Benchmark functions for the CEC'2010 special session and competition on large-scale global

optimization (2009).
[61] K. Tang, Z. Yang, T. Weise, Special session on evolutionary computation for lsgo at 2012 IEEE World Congress on Computational Intelligence, Tech. rep., University of Science and Technology of China (USTC), School of Computer Science and Technology, NICAL, Hefei, China (2012).
[62] K. Tang, X. Yao, P. Suganthan, C. MacNish, Y. Chen, C. Chen, Z. Yang, Benchmark functions for the CEC' 2008 special session and competition on LSGO, Technical report, NICAL, Dept. of Computer Science and Technology, University of Science and Technology of China, Hefei (2007).
[63] Y. Tenne, K. Izui, S. Nishiwaki, Dimensionality-reduction frameworks for computationally expensive problems, in: IEEE Congress on Evolutionary Computation, 2010, pp. 1–8.
[64] S. Thomas, Y. Jin, Reconstructing biological gene regulatory networks: where optimization meets big data, Evolutionary Intelligence 7 (1) (2014) 29–47.
[65] G. A. Trunfio, Enhancing cooperative coevolution with surrogate-assisted local search, in: X.-S. Yang (ed.), Nature-Inspired Computation in Engineering, vol. 637 of Studies in Computational Intelligence, Springer International Publishing, 2016, pp. 63–90.
[66] G. A. Trunfio, P. Topa, J. Was, A new algorithm for adapting the configuration of subcomponents in large-scale optimization with cooperative coevolution, Inf. Sci. 372 (2016) 773–795.
[67] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag New York, Inc., New York, NY, USA, 1995.
[68] Y. Wang, J. Huang, W. S. Dong, J. C. Yan, C. H. Tian, M. Li, W. T. Mo, Two-stage based ensemble optimization framework for large-scale global optimization, European Journal of Operational Research 228 (2) (2013) 308 – 320.
[69] K. Wansaseub, N. Pholdee, S. Bureerat, Optimal u-shaped baffle square-duct heat exchanger through surrogate-assisted self-adaptive differential evolution with neighbourhood search and weighted exploitation-exploration, Applied Thermal Engineering 118 (2017) 455 – 463.
[70] B. Werth, E. Pitzer, M. Affenzeller, Enabling high-dimensional surrogate-assisted optimization by using sliding windows, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17, ACM, 2017, pp. 1630–1637.
[71] R. P. Wiegand, W. C. Liles, K. A. De Jong, The effects of representational bias on collaboration methods in cooperative coevolution, in: Parallel Problem Solving from Nature — PPSN VII, LNCS, Springer, 2002, pp. 257–268.
[72] L. Willmes, T. Baeck, Y. Jin, B. Sendhoff, Comparing neural networks and kriging for fitness approximation in evolutionary optimization, in: Proceedings of IEEE Congress on Evolutionary Computation, 2003, pp. 663–670.
[73] K. Won, T. Ray, K. Tai, A framework for optimization using approximate functions, in: Proceedings of IEEE Congress on Evolutionary Computation, 2003, pp. 1077–1084.
[74] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inform. Sci. 178 (15) (2008) 2985–2999.
[75] Z. Yang, K. Tang, X. Yao, Scalability of generalized adaptive differential evolution for large-scale continuous optimization, Soft Comput. 15 (11) (2011) 2141–2155.
[76] J. Zhang, A. Sanderson, Jade: Adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[77] Z. Zhou, Y.-S. Ong, P. B. Nair, A. J. Keane, K. Y. Lum, Combining global and local surrogate models to accelerate evolutionary optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part C 37 (1) (2007) 66–76.