Information Sciences 265 (2014) 1–22

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An analysis on separability for Memetic Computing automatic design

Fabio Caraffini [a,b,1], Ferrante Neri [a,b,*,1], Lorenzo Picinali [c]

[a] Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, The Gateway, Leicester LE1 9BH, England, United Kingdom
[b] Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35 (Agora), 40014 Jyväskylä, Finland
[c] Leicester Media School, De Montfort University, The Gateway, Leicester LE1 9BH, England, United Kingdom

ARTICLE INFO

Article history:
Received 27 June 2013
Received in revised form 20 November 2013
Accepted 29 December 2013
Available online 7 January 2014

Keywords:
Memetic Computing
Optimization algorithm
Algorithmic design
Computational intelligence optimization

ABSTRACT

This paper proposes a computational prototype for automatic design of optimization algorithms. The proposed scheme makes an analysis of the problem that estimates the degree of separability of the optimization problem. The separability is estimated by computing the Pearson correlation indices between pairs of variables. These indices are then manipulated to generate a unique index that estimates the separability of the entire problem. The separability analysis is thus used to design the optimization algorithm that addresses the needs of the problem. This prototype makes use of two operators arranged in a Parallel Memetic Structure. The first operator performs moves along the axes while the second simultaneously perturbs all the variables to follow the gradient of the fitness landscape. The resulting algorithmic implementation, namely Separability Prototype for Automatic Memes (SPAM), has been tested on multiple testbeds and various dimensionality levels. The proposed computational prototype proved to be a flexible and intelligent framework capable to learn from a problem and, thanks to this learning, to outperform modern meta-heuristics representing the-state-of-the-art in optimization.

1. Introduction

Every time we have to make a decision (pick a choice) in order to achieve one or more objectives we face an optimization problem, that is the search for that feasible solution which corresponds to the extreme of an objective function. This optimization can represent various problems; for example in engineering the design of an aircraft, in economics the selection of a mid term investment, or in medicine the design of a therapy. Regardless of the nature of the application, optimization problems have a common mathematical statement. Nonetheless, optimization problems are by far not alike. Each problem is characterized by its peculiar features, such as problem dimensionality, multi-modality, ill-conditioning, separability, computational cost, and dynamic behavior.

Considering optimization theory, since the mathematical knowledge of humanity is still somewhat limited, only a very small portion of optimization problems can nowadays be approached by exact methods. Unfortunately, the non-linearities of the physical phenomena, as well as other inconveniences (e.g. a noisy objective function), make the hypotheses of the

* Corresponding author at: Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, The Gateway, Leicester LE1 9BH, England, United Kingdom. Tel.: +44 01162078570.
E-mail addresses: fcaraffini@dmu.ac.uk, fabio.caraffini@jyu.fi (F. Caraffini), fneri@dmu.ac.uk, ferrante.neri@jyu.fi (F. Neri), lpicinali@dmu.ac.uk (L. Picinali).
[1] Tel.: +358 14 260 1211; fax: +358 14 260 1021.

0020-0255/$ - see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.12.044

exact methods unverified in the majority of the real-world cases. For these reasons, and thanks to the diffusion of powerful computers, many heuristic methods have been proposed and implemented in order to tackle optimization problems.

For about three decades, between 1960s and 1990s, computer scientists have investigated optimization methods with the aim of detecting the ultimate algorithm capable of outperforming all the others. During those years, important computational paradigms such as Simulated Annealing, and Evolutionary and Swarm Intelligence Algorithms (see for example [1]), were defined.

The search for the best optimizer has been stopped by the publication of the No Free Lunch Theorems (NFLT) [2]. The NFLT mathematically prove that the average performance of any pair of algorithms A and B across all possible problems is the same. This fact implies that if an algorithm performs well on a certain class of problems, then it necessarily pays for that with degraded performance on the set of all remaining problems (as this would be the only way that all algorithms can have the same performance averaged over all the optimization problems). The NFLT has been proven only for discrete sets and under the (often not occurring) hypothesis that the algorithms are non-revisiting, i.e. the algorithms do not perform the objective function evaluation of the same candidate solution more often than once during the optimization run. Thus, the NFLT is not generally valid, as discussed in [3,4]. Nonetheless, the concept that there is no universal optimizer had a significant impact on the scientific community. More specifically, since the performance of each algorithm over all the possible problems has theoretically been proven to be the same, it was clear that there was no longer a reason to discuss which algorithm is, in general, better or worse. On the contrary, as the most direct consequence of NFLT, computer scientists started to propose algorithms which were tailored to specific problems in order to design an optimizer that could display its highest performance for the specific problem of interest, e.g. see [5–7].

In real-world problems, the use of a tailored algorithm (with its parameter setting) for each problem could be extremely impractical as it would likely require the constant action of an optimization expert to adjust the design to the industrial needs. In addition, the implicit non-stationary nature of optimization processes imposes that there is no unique most suitable algorithm and parameter setting during the run-time. In order to tackle these issues, modern optimization algorithms are composed of multiple search operators that are coordinated with the aim of promptly reacting to the necessities of the optimization processes and thus successfully address reasonable ranges of problem features. Although a proper taxonomy cannot be easily formulated, due the overlap of the various classes, we here propose a classification of modern optimization algorithms:

- **Adaptive and self-adaptive algorithms based on a paradigm.** These algorithms are based on one single paradigm/optimization framework, but include different variants of it. The search operators corresponding to each variant are coordinated by means of a feedback based criterion. For example, in the context of Differential Evolution (DE), the algorithm proposed in [8] combines multiple mutation strategies coordinated on the basis of a learning period and a randomized success based logic. In [9] a so called *ensemble* of mutation and crossover strategies, as well as the related parameters are encoded within the solutions and evolve with them. Other harmonic self-adaptive combinations of components within the DE framework are proposed in [10,11]. In [12] an adaptive memetic approach is designed in the context of Particle Swarm Optimization (PSO) with reference to DNA sequence compression.
- **Hyper-heuristics and portfolio algorithms.** In these frameworks, a list of multiple algorithms is coordinated by means of a heuristic rule or supervisory/adaptive scheme. Some examples of portfolios of heuristically coordinated algorithms have been proposed in [13,14]. Another popular portfolio approach for the propositional satisfiability problem is the so called SATzilla platform, see [15,16]. Within this platform, a study to assess the trade-off of the search algorithms with the aim at determining an automatic coordination system is presented in [17]. A recent study that models the behavior of optimizers and predicts their run time is given in [18]. In hyper-heuristics the coordination amongst multiple algorithms is performed by means of a machine learning technique. The latter acts as a supervisor structure that learns which algorithms are the most suitable for a given problem. A famous example of hyper-heuristic implementation in the field of timetabling and rostering are proposed in [19,20] while two extensive literature reviews on the topic are given in [21,22], respectively. In [23] graph coloring heuristics are coupled with a random ordering heuristic. Several attempts to perform the coordination of the heuristics are given in the literature. A classic approach consists of assigning a score and rewarding the most promising heuristics: the so called choice function, see [19]. A combination of the choice function with a randomized criterion is also very broadly used, see e.g. [24]. Among many coordination schemes present in the literature, more sophisticated approaches make use of reinforcement learning in a stand alone or combined fashion, see e.g. [20,25], and memory-based mechanisms, see [26]. In [27,28] elegant learning schemes are coupled with multiple operators (multi-agents) for addressing complex optimization problems.
- **Memetic Computing Structures.** Memetic Computing (MC) is a subject that studies optimization algorithms which are structures composed of interacting operators, see for example [29,30]. Originally, the word Memetic Algorithm (MA) was used to indicate hybrid algorithms [31] composed of an evolutionary framework and one or more local search within the generation cycle, see [32,33]. MC extends and generalizes the idea of algorithms composed of multiple operators to something that does not fix/specify any structure before an algorithmic analysis of the problem is performed. In MC, an evolutionary framework is not mandatory as an efficient algorithm (for a given problem) can be based on multiple subsequent local searches [34] or even by operators perturbing a single solution [35].

In some cases this classification is mainly a philosophical and cannot be easily spotted within the implementation details. For example, some MC structures can also be adaptive, see [36–40]. Nonetheless, this philosophical distinction has a major impact when we consider the future trend of research in Computational Intelligence (CI). More specifically, the first category attempts to adapt to a range of various problems single algorithmic paradigms. The second category tackles optimization problems by means of a proper selection of algorithms from a database. The third category sees each algorithm as a structured collection of operators (memes), and each problem as the starting point for a new algorithmic design. This philosophical feature of MC contains a visionary potential for the future in optimization, see [41,42]. More specifically, if the design of an optimization algorithm consists of selecting its suitable operators from a pool as well as their interaction relationship, i.e. the links between the operators, this bottom-up design allows a straightforward implementation within a computational device, without relying on human experience, see [35].

The topic of automatic design of optimization algorithm is currently intensively discussed within the MC community, see [43,44], as well as in related fields of computational sciences where similar concepts and issues are analyzed from slightly different perspectives, see e.g. [45–47]. Automatic design of optimization algorithms is thoroughly studied in [48] from multiple perspectives. For example, in [16] a survey on automatic parameter setting is given while in [49] a learning system for the coordination of heuristics in the context of constrained optimization is presented. In [50] the use of reinforcement learning for online parameter tuning of stochastic local search algorithms is shown. In [51] an evolutionary framework that adaptively selects its components during the run time is presented. In [52] an adaptation technique to improve the performance of a component of the SATzilla platform is shown.

Moreover, a software package namely FRACE, illustrated in[53], employs a statistical procedure for automatically performing the configuration of the algorithms. An extension of FRACE, namely IRACE, is presented in [54]. However, due to its complexity, an efficient and generally valid solution has not been found yet and, in our opinion, will require still several years of research in mathematics and computer science. It must be remarked that this topic has a major impact within the entire Artificial Intelligence (AI) field in the mid and long terms. More specifically, a machine that is able to "understand" what is the suitable solver in every circumstance is a machine capable to critically make decisions. In other words, since intelligence can be seen as the talent of performing on a regular basis the "right" choice, a machine that builds up by itself the reasoning to perform the best choice has a level of automation superior to that present currently in computational machines.

This paper proposes a procedure for automatizing the design of MC structures (and, in general, of optimization algorithms) and for the implementation of a fully functional computational prototype for continuous optimization problems. The general proposed procedure consists of two phases. At first a problem analyser detects the features of the problem. Subsequently these features are used to select the operators and their links thus performing the algorithmic design automatically. The implemented problem analyser estimates the separability of the problem and extracts an index indicating the so-called *degree of separability*, see SubSection 3.1. This index is then used to balance the action of an operator that moves along the axes with respect to another operator that performs diagonal moves by estimating the local gradient. The proposed implementation will be here referred as Separability Prototype for Automatic Memes (SPAM).

The remainder of this paper is organized in the following way. Section 2 describes the elementary MC structures. Section 3 gives description and motivation for the procedure to analyze the separability of the problem. Section 3 also shows how the information about the separability is used to automatically design the algorithm. Section 4 displays the numerical results computed over multiple testbeds and dimensionality levels against eight modern algorithms representing the state-of-the-art in continuous optimization. Finally, Section 5 gives the final remarks of this work.

2. Background: elementary Memetic Computing Structures

Without a loss of generality, in order to clarify the notation in this paper, we refer to the minimization problem of an objective function (or fitness) $f(\mathbf{x})$, where the candidate solution $\mathbf{x}$ is a vector of n design variables (or genes) in a decision space $\mathbf{D}$. Thus, the optimization problem considered in this paper consists of the detection of that solution $\mathbf{x}^* \in \mathbf{D}$ such that $f(\mathbf{x}^*) < f(\mathbf{x})$, and this is valid $\forall \mathbf{x} \in \mathbf{D}$. Array variables are highlighted in bold face throughout this paper.

Optimization algorithms can be seen, at an abstract level, as mathematical procedures that address optimization problems by combining operations of two kinds: search operations where one or more solutions (a population) are generated/computed within the search space, and selection operations, when one or more solutions are selected and retained, see [55]. This definition is valid regardless of the fact the algorithm is an exact or a heuristic method. In general, we can consider all the algorithms as population-based where single-solution algorithms are a special case of them (population size 1). At the generic step t, a set of candidate solutions are stored into a memory structure, namely population $\mathbf{Pop^t}$, which contains the current solutions. In order to improve upon the available candidate solutions, a sub-operator U processes the population $\mathbf{Pop^t}$ and returns a new population of (trial) candidate solutions **Trials** after having applied the search logic of the algorithm:

$$\mathbf{Trials} = U(\mathbf{Pop^t}). \tag{1}$$

Subsequently, the selection sub-operator S processes both the populations $\mathbf{Pop^t}$ and **Trials** and returns a new current population for the step $t+1$:

$$\mathbf{Pop}^{t+1} = S(\mathbf{Pop}^{t}, \mathbf{Trials}). \tag{2}$$

The selection sub-operator S obviously requires the fitness evaluation of the candidate solutions. For evolutionary algorithms, the above-mentioned formulas are already explicitly formulated as U is the application of variation operators (crossover and mutation) while S is the selection strategy which can be, for example, the selection of parents and generational replacement for classical GAs or the so called "plus" strategy in Evolution Strategies (ES), see [1].

Within MC view, paired sub-operators U and S (for a certain amount of steps t) compose an algorithmic operator, while optimization algorithms are seen as connected structures of operators. Each operator is associated with an input and output **Pop**. MC structures can be complex and be composed of Elementary Memetic Structures (EMSs). The first EMS is called *Sequential Memetic Structure* and consists of multiple operators that sequentially process a population **Pop**, i.e. the output of an operator is univocally the input of the following operator, see [35]. Fig. 1 gives a graphical representation of a sequential structure.

Instead of having a univocal path, the output population from an operator could be further processed by a decisional component that selects the subsequent operator which will attempt to improve upon **Pop**. The selection criteria can be based on a probability or a fitness-based rule (e.g. the success of the previous operator). This decisional component is called *Memetic Node* (MN) and is defined as a decisional mechanism that allows the selection of one meme (operator) amongst multiple choices, see [56]. A graphical representation of MN is given in Fig. 2. Analogous to an electrical node, a MN can be seen as the connection of at least three memes/operators/components; one is the operator that has just been activated, and two (or more) of them represent the list of memes from which the subsequent operator should be selected. Operators that concur to the same MN are said to be *in parallel*. A sub-structure containing only one MN is here called *Parallel Memetic Structure*, and this is the second EMS here taken into account. Fig. 3 shows a graphical representation of a Parallel Memetic Structure where operators 2, 3, and 4 are in parallel, and the sub-structure composed of the MN, operators 2, 3, 4 and respective links is a parallel structure.

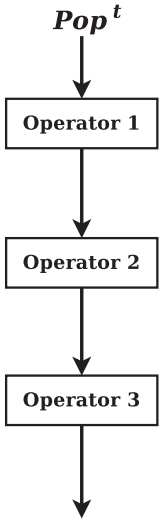

Fig. 1. Graphical representation of Sequential Memetic Structure.

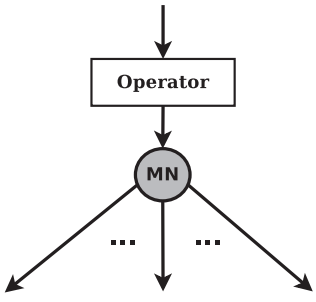

Fig. 2. Graphical representation of Memetic Node.

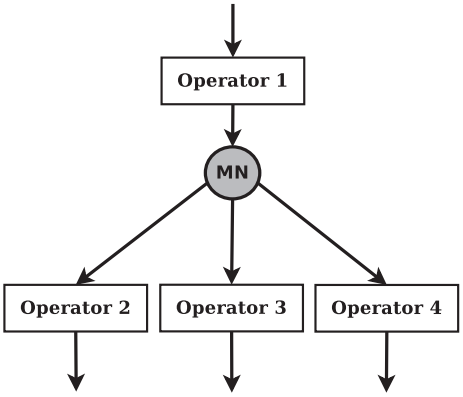

Fig. 3. Graphical representation of Parallel Memetic Structure.

3. Separability prototype for automatic memes

Before entering into the implementation details of the proposed SPAM algorithm, a description of the general idea and software platform is given. The algorithmic design is automatically performed by the machine after a problem analysis. More specifically, an initial portion of the budget is used to analyze the problem and extract its features. For each feature, an index will be assigned. These indices will be then used for selecting and combining the proper operators. A graphical representation of the general idea of the automatic design platform is given in Fig. 4 The present implementation is a prototype restricted to the separability. In other words, this paper proposes a first component of this problem analyser and an example of MC implementation based on the above mentioned problem analysis approach.

3.1. Separability analysis

A function f of n independent variables is said to be separable if it can be expressed as a sum of n functions, each of them depending on one variable only. This definition implies that, from an optimization point of view, separable functions are relatively easy to handle as the optimization problem in n variables can be tackled efficiently by perturbing separately each variable. However, real-world applications are often (if not always) characterized by non-separable fitness functions, i.e. functions in which there is some degree of non-linear inter-variable interaction. On the contrary, non-separable problems cannot be solved by performing moves along the axes but require simultaneous perturbations of multiple variables (diagonal moves). While, according to rigorous mathematics, an interaction between a pair of variables only is enough to make the problem non-separable, in computational science this is not entirely true. According to the number of interacting variables, a function (and thus the associated problem) can be considered separable to a certain degree, being e.g. fully separable, moderately separable, moderately non-separable, or fully non-separable. Modern testbeds tend to classify test problems according to similar criteria, e.g. see [57]. On the basis of this consideration, an estimation of the degree of separability (in a fuzzy logic fashion) has here been defined. A set of λ candidate solutions is sampled within $\mathbf{D}$. For a limited portion of the budget the Covariance Matrix Adaptation Evolution Strategy (CMAES) with rank-μ-update and weighted recombination, see [58], is applied. This optimization method consists of sampling from a multivariate distribution λ point, compute their fitness values and update the shape of the distribution in order to progressively adapt to the basins of attraction. The sampling of the individual k at the generation $g+1$ is given by:

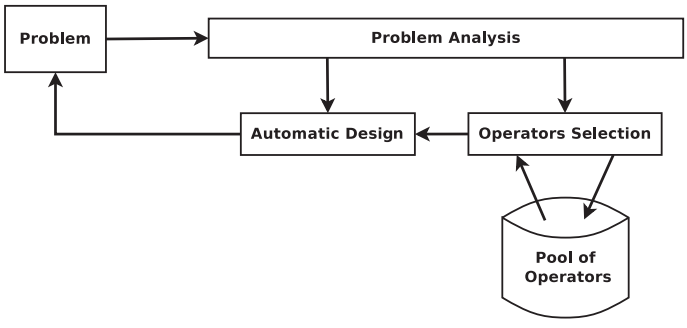

Fig. 4. Graphical representation of the general idea of automatic design platform with problem analyser.

$$x_k^{(g+1)} \sim \mathcal{N}\left(\langle \mathbf{x}\rangle_w^g, (\sigma^g)^2 \mathbf{C}^g\right) \tag{3}$$

where $\mathcal{N}\ (m, \sigma^2 C)$ is a multivariate normal distribution of mean m, stepsize σ, and estimated covariance matrix $\mathbf{C}$. The mean value $\langle \mathbf{x}\rangle_w^g$ is a weighted sum of the μ candidate solutions ($\mu \leqslant \lambda$) displaying the best performance at the generation g (those individuals that are associated to the lowest fitness values $f(\mathbf{x})$). This vector corresponds to a recombination result, see [58] for details. Stepsize σ and covariance matrix are progressively updated at each generation. The update rule for the covariance matrix is:

$$\mathbf{C}^{g+1} = (1 - c_{cov})\mathbf{C}^g + c_{cov}\frac{1}{\mu_{cov}}\mathbf{p}_\mathbf{c}^{g+1}\left(\mathbf{p}_\mathbf{c}^{g+1}\right)^T + c_{cov}\left(1 - \frac{1}{\mu_{cov}}\right)\sum_{i=1}^{\mu}\left(\mathbf{x}_{i-b}^{g+1} - \langle \mathbf{x}\rangle_w^g\right)\left(\mathbf{x}_{i-b}^{g+1} - \langle \mathbf{x}\rangle_w^g\right)^T$$

where $\mathbf{x}_{i-b}^g$ denotes the ith best individuals at the generation g, c_{cov} is a parameter determining the learning rate for the estimated covariance matrix $\mathbf{C}$, and $\mathbf{p}_\mathbf{c}$ is a vector namely evolution path that determines the adaptation of the covariance matrix. The update formula is given by:

$$\mathbf{p}_\mathbf{c}^{g+1} = (1 - c_c)\mathbf{p}_\mathbf{c}^g + H_\sigma^{g+1}\sqrt{c_c(2 - c_c)}\frac{\sqrt{\mu_{eff}}}{\sigma^g}$$

$$\left(\langle \mathbf{x}\rangle_w^{g+1} - \langle \mathbf{x}\rangle_w^g\right)$$

where $\mu_{eff} = \frac{1}{\sum_1^\mu w_i^2}$, c_c is a parameter, H_σ^{g+1} is a function defined by cases that can take values 0 or 1. Also the stepsize σ^{g+1} is iteratively updated. Details about CMAES implementations can be found e.g. in [59,60,58]. At the end of each generation the μ individuals displaying the best performance are selected and used to compute $\langle \mathbf{x}\rangle_w^{g+1}$.

After a certain amount of generations, the matrix $\mathbf{C}$ evolves and reliably approximates the (theoretical) covariance matrix. A covariance matrix is a correlation matrix, i.e. a matrix that describes the correlation between pairs of variables. For the estimated covariance matrix $\mathbf{C}$, indicating the generic term of the matrix $C_{i,j}$, the following matrix transformation is applied:

$$\rho_{i,j} = \frac{C_{i,j}}{\sqrt{C_{i,i}C_{j,j}}}. \tag{4}$$

The operation in Eq. (4) describes the Pearson correlation coefficient and the matrix ρ is the Pearson correlation matrix. These coefficients vary between -1 and 1 and measure the linear correlation between pairs of variables. When $\rho_{i,j} = 0$ there is no correlation at all between the ith and jth variables. When $|\rho_{i,j}| = 1$ there is a perfect correlation between the variables. More specifically, when $\rho_{i,j} = 1$ it means that to an increase of the ith variable corresponds the same (linear) increase of the jth variable, when $\rho_{i,j} = -1$, it means that to an increase of the ith variable corresponds the same (linear) decrease of the jth variable. The Pearson correlation matrix has been chosen instead of directly the covariance matrix because its elements are limited and normalized within the $[-1,1]$ interval, and thus allows an immediate interpretation for the purposes of this problem analyser.

If there is no correlation between any pair of variables, and the corresponding optimization problem can be solved by perturbing each variable separately. Conversely, if the variables are correlated, search moves in optimization require a simultaneous perturbation of multiple variables. Thus, as shown in [61], although there is no rigorous mathematical equivalence, the Pearson correlation matrix ρ can be viewed as a description of the separability of the optimization problem.

In order to use this description with the aim of designing an algorithm, the absolute value of the Pearson correlation matrix $|\rho|$ is computed as there is no interest in distinguish between positive and negative correlation, since they would both result into the application of a simultaneous perturbation of the variables. The resulting matrix is symmetrical and exhibits ones on the diagonal:

$$|\rho| = \begin{vmatrix} 1 & |\rho_{1,2}| & |\rho_{1,3}| & \dots & |\rho_{1,n}| \\ X & 1 & |\rho_{2,3}| & \dots & |\rho_{2,n}| \\ X & X & 1 & \dots & |\rho_{3,n}| \\ \dots & \dots & \dots & \dots & \dots \\ X & X & X & X & 1 \end{vmatrix}.$$

Thus, only $\frac{(n^2-n)}{2}$ elements of the matrix $|\rho|$ are of interest for evaluating the degree of separability of the problem. In order to extract an index that estimates the separability of a problem, an average of the elements of the matrix $|\rho|$ is computed. However, since the initial budget to evolve the matrix $\mathbf{C}$ is limited, the Pearson correlation matrix normally contains approximation errors. These errors could jeopardize the reliability of the separability index. In order to mitigate this effect, the matrix $|\rho|$ is processed by a rounding procedure that makes correspond to 0 a $|\rho_{i,j}| \in [0, 0.2[$, to 0.3 a $|\rho_{i,j}| \in [0.2, 0.4[$, to 0.5 a $|\rho_{i,j}| \in [0.4, 0.6[$, to 0.5 a $|\rho_{i,j}| \in [0.6, 0.8[$, and to 1 a $|\rho_{i,j}| \in [0.8, 1]$. The discrete values resulting from this process are indicated as $\rho_{i,j}^d$. A graphical scheme representing this rounding procedure is shown in Fig. 5. Finally, the average of these values is calculated:

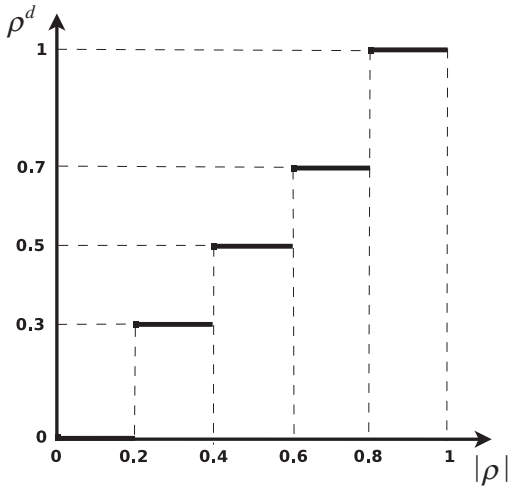

Fig. 5. Rounding procedure of the correlation coefficients.

$$\varsigma = \frac{2}{(n^2 - n)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \rho_{ij}^{d}. \tag{5}$$

Thus, the separability index ς is an estimation of the degree of separability of the optimization problem under examination. When this index $\varsigma = 0$, the problem is considered fully separable; when $\varsigma = 1$, the problem is considered fully non-separable; in the remaining cases the problem is considered to have intermediate features.

In order to experimentally demonstrate the validity of the proposed approach. The popular IEEE CEC2005 testbedin 30 dimensions, see [62], has been considered. For each problem contained in the testbed the problem analysis described above has been performed and the separability index ς has been calculated. Table 1 displays, for each optimization problem, the corresponding ς value and the indication on separability originally reported in [62].

As shown in Table 1, the proposed coefficient ς reliably estimates the separability of problems f_1, f_9, and f_{15}. In the remaining cases, the value of the index ς is on average larger than in the separable cases. In addition it can be seen that the separability degree varies over the problem. While in the case of a sphere (f_1) the problem is clearly separable and thus the

Table 1
Separability Index (ς) for CEC2005 in 30 dimensions.

	Separability index (ς)	Separable
f_1	0.043	YES
f_2	0.080	–
f_3	0.399	–
f_4	0.189	–
f_5	0.226	–
f_6	0.104	–
f_7	0.067	–
f_8	0.195	–
f_9	0.058	YES
f_{10}	0.257	–
f_{11}	0.183	–
f_{12}	0.672	–
f_{13}	0.404	–
f_{14}	0.510	–
f_{15}	0.046	YES
f_{16}	0.121	–
f_{17}	0.186	–
f_{18}	0.1638	–
f_{19}	0.143	–
f_{20}	0.165	–
f_{21}	0.073	–
f_{22}	0.175	–
f_{23}	0.155	–
f_{24}	0.050	–
f_{25}	0.190	–

index $\varsigma \approx 0$, in other cases, such as f_2 the problem (Shifted Schwefels Problem 1.2) is only weakly non-separable. It can be observed that in f_2 it is mathematically non-separable, but can still be optimized by taking into consideration the variables one-by-one. More specifically, the mathematical expression of f_2 is (apart from shift and bias) of the form $\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$. Although being non-separable, this problem can be solved by optimizing at first x_1^2 (and obtaining 0), then substituting in $x_1^2 + x_2^2$ and solving with respect to x_2 and so on. If we are under the hypotheses that the problem is a black box and thus we ignore its analytical properties, this problem greatly benefits from an algorithm that perturbs the variables one by one. In other words, from an optimization perspective this problem is "almost separable". Another interesting case is f_{24}, that albeit being a complex composition function, it appears to be nearly separable. While it is not immediate to show that this function can be optimized by considering the variables separately, numerical results will show that the application of search moves along the axes lead to an excellent performance.

It can be observed that the separability and the problem of correlation among design variables is a major problem in optimization. Besides, the works related to the covariance matrix adaptation and multiple versions of CMAES, it is worth mentioning the linkage learning in evolutionary algorithms. The linkage learning is the process of grouping together the sets of design variables that are functionally correlated. Thus, in optimization linkage learning can be seen as a technique to decompose an optimization problem so that instead of a large scale non-separable problem, multiple low dimensional non-separable problems can be tackled. This idea is used originally for guiding the offspring generation in Genetic Algorithms, see [63,64], and used in Estimation of Distribution Algorithms, see [65], as well as compact optimization, see [66,67]. Our future works may consider linkage learning as a basic technique for grouping the variables in the context of an analysis of separability.

3.2. The pool of operators

The SPAM algorithm makes use of a pool of two operators. The same operators used in [56] are considered in this work.

The first operator, here indicated with S, is a local search algorithm which perturbs a single solution along its n axes, i.e. separately perturbs each design variable. This meme can be seen as a modification of a classical hill-descend algorithm and was used for the first time in [68] within a memetic structure.

The S implementation requires a generic input solution $\mathbf{x}$ and a trial solution $\mathbf{x^t}$. S perturbs the candidate by computing, for each coordinate i (each gene),

$$x_i^t = x_i - \xi, \tag{6}$$

where ξ is the exploratory radius. Subsequently, if $\mathbf{x^t}$ outperforms $\mathbf{x}$, the solution $\mathbf{x}$ is updated (the values of $\mathbf{x^t}$ are copied in it), otherwise a half step in the opposite direction is performed:

$$x_i^t = x_i + \frac{\xi}{2}. \tag{7}$$

Again, $\mathbf{x^t}$ replaces $\mathbf{x}$ if it outperforms it. If there is no update, i.e. the exploration was unsuccessful, the radius ξ is halved. This exploration is repeated for all the design variables. For the sake of clarity, Algorithm 1 describes the working principles of the S operator.

Algorithm 1. Pseudo-code of the S operator.

```
INPUT x
while condition on the local computational budget do
  for i = 1 : n do
    x_i^t = x_i − ξ
    if f(x^t) ⩽ f(x) then
      x = x^t
    else
      x_i^t = x_i + ξ/2
      if f(x^t) ⩽ f(x) then
        x = x^t
      end if
    end if
  end for
  if x == x^t then
    ξ = ξ/2
  end if
end while
OUTPUT x
```

The second operator is the Rosenbrock algorithm (R), see [69]. This operator, has been proved to always converge to a local optimum, under specific conditions, see [70]. At the beginning of the optimization of this component, R is similar to S as it explores each of the n directions, perturbing the input solution $\mathbf{x}$ with an initial step size vector $\mathbf{h}$. A matrix $\mathbf{A}$ is also initialized as the identity matrix. Each step of this algorithm consists of the following. As long as new improvements are found, for $j = 1, 2, \ldots, n$, a new trial point $\mathbf{x^t}$ is generated perturbing the each ith design variable of solution $\mathbf{x}$ in the following way:

$$x_i^t = x_i + h_j \cdot A_{i,j} \tag{8}$$

for $i = 1, 2, \ldots, n$. In case of success (the trial solution outperforms the solution $\mathbf{x}$), $\mathbf{x}$ is updated and the step size is increased of a factor α ($h_j = \alpha \cdot h_j$), otherwise it is decreased by means of a factor β and the opposite direction is tried ($h_j = -\beta \cdot h_j$). As said, this procedure is repeated until it is possible to improve upon the solution $\mathbf{x}$. Once every possible success has been found and exploited in each base direction, the coordinate system is rotated towards the approximated gradient by means of the Gram-Schmidt orthogonalization procedure. This operation results into an update of the matrix $\mathbf{A}$. After the orthogonalization, the step size vector $\mathbf{h}$ is reinitialized and the procedure is repeated, using the rotated coordinate system, perturbing again the current elite $\mathbf{x}$ according to Eq. (8): it is important to notice that, when a rotated coordinate system is used, i.e. the matrix $\mathbf{A}$ is no longer an identity matrix, this trial generation mechanism corresponds to a diagonal move by following the direction given by the gradient. The Rosenbrock Algorithm terminates when a stop criterion is met. The stop criterion is given by two conditions. The first criterion is based on the minimum element of the perturbation vector $\mathbf{h}$, the second is based on the minimum difference between $\mathbf{x^t}$ and $\mathbf{x}$ design variables. More specifically, R is continued until the following condition is true:

$$\min(|\mathbf{h}|) > \varepsilon \; OR \; \min(|\mathbf{x^t} - \mathbf{x}|) > \varepsilon \tag{9}$$

where min() is the minimum vector element. If for a step there is no improvement at all, only the first condition is considered. In summary SPAM makes use of two very diverse operators in terms of search logic. While S moves along the axes and attempts to search for fitness improvements by perturbing the variables one by one, R perturbs all the variables at the same time by following the local gradient. Clearly, S and R are here thought for handling separable and non-separable problems, respectively. Graphical representations of the S and R search moves are shown in Figs. 6 and 7.

3.3. Automatic memetic design

When the index ς is available, the automatic design is performed. The SPAM algorithm makes use of a Parallel Memetic Structure where two operators in parallel alternatively perturb a single solution. The best solution ever found after the application of each operator is here indicated as *elite* $\mathbf{x^e}$. With reference to the abstract analysis of algorithms reported in Section 2, the elite plays the role of the population **Pop** (a population of only one individual in this case), while the **Trials** is the trial solution $\mathbf{x^t}$ mentioned above. The initial solution is the best solution detected during the problem analysis, i.e. the best solution obtained by CMAES with a limited budget. The index ς is used to assign, for each problem, an activation probability within the MN. If $\varsigma = 0$, the problem is considered separable and only S is repeatedly used to optimize the function. If ς is high ($\varsigma > 0.5$), the problem is considered fully non-separable and only R is repeatedly used to optimize the problem. In the remaining cases, the problem is considered to have intermediate features and the two operators coexist (or compete and collaborate). In the latter case, the assignment of the two probabilities are given by a linear trend. Fig. 8 illustrates the activation probabilities of the two trends.

Finally, due to the deterministic nature of the operators, in order to avoid that one operator is called to perform the same steps in vain, a control has been implemented. If one operator failed at improving upon $\mathbf{x^e}$ and is selected again by the MN, the optimization is not performed on $\mathbf{x^e}$ but on another solution $\mathbf{x^p}$ resulting from a stochastic perturbation of $\mathbf{x^e}$. More

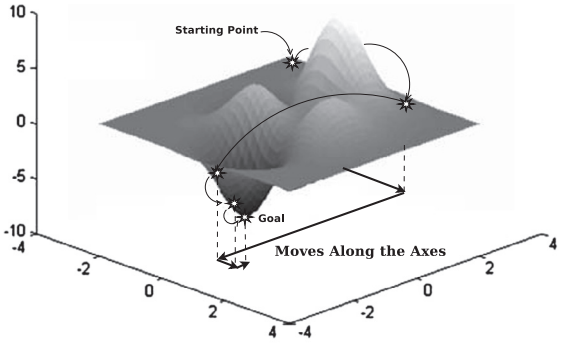

Fig. 6. Graphical representation of S search logic.

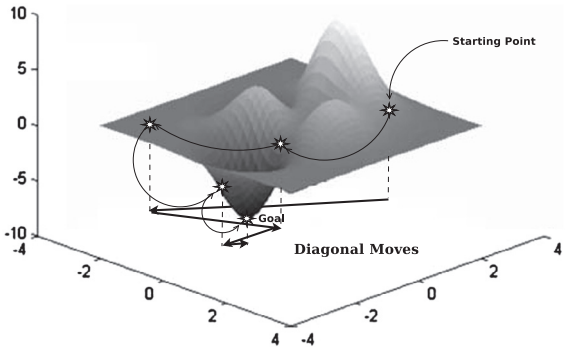

Fig. 7. Graphical representation of R search logic.

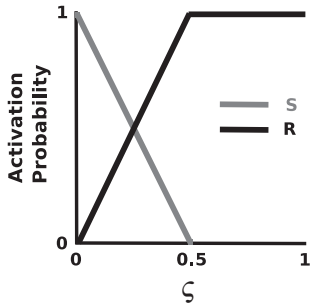

Fig. 8. Activation probabilities of the operators.

specifically, at first a new solution $\mathbf{x^r}$ is randomly sampled within $\mathbf{D}$. Then, the perturbed solution $\mathbf{x^p}$ is composed of the elite $\mathbf{x^e}$ with a section of it replaced by a section copied from $\mathbf{x^r}$. This section replacement occurs according to the rules of a DE exponential crossover, see e.g. [71]: one gene is randomly selected from $\mathbf{x^r}$ and copied into $\mathbf{x^e}$ while the subsequent genes are copied from $\mathbf{x^r}$ to $\mathbf{x^e}$ until newly generated random numbers are smaller than a threshold CR (that stands for crossover rate). A pseudo-code of this mechanism is given in Algorithm 2.

Algorithm 2. Pseudo-code of the perturbation mechanism.

INPUT $\mathbf{x^e}$
$\mathbf{x^p} = \mathbf{x^e}$
randomly generate $\mathbf{x^r}$ within the decision $\mathbf{D}$
randomly generate an integer number i between 1 and n
$x_i^p = x_i^r$
$k = 1$
while $rand(0,1) \leqslant CR$ AND $k < n$ **do**
  $x_i^p = x_i^r$
  $i = i + 1$
  if $i == n$ **then**
    $i = 1$
  end if
  $k = k + 1$
end while
OUTPUT $\mathbf{x^p}$

As a further remark, although we indicate, for the aim of using a clear notation, the perturbed solution as $\mathbf{x^p}$, it can be seen as a **Trials** data structure. As shown in [72], it can easily be observed that for a given value of CR, the effect of the exponential crossover changes with the dimensionality of the problem. For low-dimensional problems, the perturbed solution will be dramatically perturbed, while for higher dimensionality values the solution will be only moderately perturbed. In order to avoid this problem and make the crossover action independent on the dimensionality of the problem, the following quantity, namely inheritance factor, is fixed:

$$\alpha_e \approx \frac{n_e}{n} \tag{10}$$

where n_e is the number of genes we expect to copy from $\mathbf{x^r}$ into $\mathbf{x^p}$ in addition to the first gene, which is deterministically copied. The probability that n_e genes are copied is $Cr^{n_e} = Cr^{n\alpha_e}$. In order to control the approximate amount of copied genes and to achieve that about n_e genes are copied into the offspring with probability 0.5, we impose that:

$$Cr^{n\alpha_e} = 0.5. \tag{11}$$

It can easily be seen that, for a chosen α_e, the crossover rate can be set on the basis of the dimensionality as follows:

$$Cr = \frac{1}{\sqrt[n\alpha_e]{2}}. \tag{12}$$

By means of formula (12), the expected that the degree of perturbation (amount of perturbed genes) is thus controlled. The pseudo-code displaying the working principles of the proposed SPAM algorithm as well as its phases is given in Algorithm 3.

Algorithm 3. Pseudo-code of the SPAM algorithm.

```
*** PROBLEM ANALYSIS ***
INPUT problem f(x) in a continuous domain D
while the budget of the problem analysis is available do
    perform separability analysis
end while
OUTPUT the elite x^e and the separability index ς
*** AUTOMATIC DESIGN ***
INPUT the separability index ς and operators S and R
compute the activation probabilities for S and R within MN on the basis of ς, see Fig. 8
OUTPUT activation probabilities for S and R
*** OPTIMIZATION ***
INPUT the elite x^e and the activation probabilities for S and R
x^p = x^e
generate a random number rand(0,1) for selecting the operator
while the remaining budget is available do
    select the operator (S or R) on the basis of rand(0,1) and apply it to x^p
    if the operator succeeded at improving upon x^e performance then
        x^e = x^p
    end if
    generate a random number rand(0,1) for selecting the operator
    if the operator failed at improving upon x^e performance AND the selected operator is the same that failed then
        compute the perturbed x^p according to Algorithm 2
        if x^p has a better performance than that of x^e then
            x^e = x^p
        end if
    end if
end while
OUTPUT Improved elite solution x^e
```

4. Numerical results

In order to test the flexibility and potentials of the SPAM algorithm, it has been tested on the following testbeds:

- The CEC2013 benchmark described in [73] in 10 dimensions (28 test problems).
- The CEC2005 benchmark described in [62] in 30 dimensions (25 test problems).
- The CEC2013 benchmark described in [73] in 50 dimensions (28 test problems).
- The BBOB2010 benchmark described in [57] in 100 dimensions (24 test problems).
- The CEC2008 benchmark described in [74] in 1000 dimensions (7 test problems).
- The CEC2010 benchmark described in [75] in 1000 dimensions (20 test problems).

We are intentionally considering many test problems (totally 132) in order to justify how SPAM adapts to problems and is able to tackle them. All the results in this paper are presented in terms of error. Since the theoretical minimum of f_7 problem

of CEC2008 is unknown, in order to avoid confusions, its result is not displayed but is included in the statistical test (se Holm-Bonferroni procedure below).

For each algorithm in this paper (see following subsections) 100 runs have been performed for each of the 132 problems. Each run has been continued for $5000 \times n$ fitness evaluations, where n is the dimensionality of the problem. For each test problem and each algorithm, the average final fitness value ± standard deviation over the 100 available runs has been computed. In order to strengthen the statistical significance of the results, for each test problem the Wilcoxon Rank-Sum Test [76] has been also applied, with a confidence level of 0.95. A + indicates that the proposed SPAM algorithm significantly outperforms the competitor, a − indicates that SPAM is outperformed, and a = is shown when the performance of the two algorithms is statistically indistinguishable.

Throughout this paper, SPAM parameters are those of S and R. Regarding S, the initial search radius ξ, it has been set equal to 40% of the domain width. Since we are using hyper-cubical decision spaces, this parameter is a scalar (more precisely it is a vector exhibiting the same number). In the case of a hyper-rectangle, the search radius would be a vector. Regarding R, h is a vector of $0.1, \alpha = 2, \beta = 0.5$, while ε in Eq. (9) has been set equal to 10^{-5}. As for the problem analyser, the standard internal parameter setting of CMAES has been used, see [77].

4.1. Does the automatic design actually work? An experimental test

In order to experimentally demonstrate the effectiveness of the automatic design, SPAM has been compared with an algorithm composed of the same operators, that performs the analysis (thus detecting the initial solution $\mathbf{x_e}$) but then forces the activation probabilities of S and R to 0.5. This algorithm, namely SPAM$^{0.5}$ is a version of SPAM without the automatic design based on separability analysis. Numerical results of this test are shown in Tables 2–7. In each table, the average error ± standard deviation (over the 100 runs available) and Wilcoxon Rank-Sum Test are shown. Numerical results clearly show that the proposed probability setting based on the analysis of separability leads to a performance equally high or higher than the equally distributed one. We observed that SPAM and SPAM$^{0.5}$ display a similar performance when the separability analysis lead to a 0.5 and 0.5 setting of S and R probabilities, i.e. when the two algorithms coincide. If we consider that this experiment has been carried on a diverse testbed containing 132 problems, the success in the totality of comparison gives a convincing indication that the proposed framework is versatile and efficient.

4.2. Comparison against the-state-of-the art algorithms

In order to test the potentials of SPAM with respect to modern algorithms in literature, we selected the following three new and efficient meta-heuristics:

Table 2
SPAM against SPAM$^{0.5}$ on CEC2013 in 10D.

	SPAM	SPAM$^{0.5}$	
f_1	**0.00e + 00 ± 0.00e + 00**	0.00e + 00 ± 0.00e + 00	=
f_2	**0.00e + 00 ± 0.00e + 00**	0.00e + 00 ± 0.00e + 00	=
f_3	**4.45e + 01 ± 2.61e + 02**	8.36e + 02 ± 7.47e + 03	=
f_4	**0.00e + 00 ± 0.00e + 00**	0.00e + 00 ± 0.00e + 00	=
f_5	**2.36e − 08 ± 2.77e − 08**	5.21e − 08 ± 8.76e − 08	+
f_6	**3.94e + 00 ± 4.67e + 00**	6.70e + 00 ± 6.80e + 00	+
f_7	**6.67e + 01 ± 5.65e + 01**	2.22e + 09 ± 1.94e + 10	=
f_8	**2.03e + 01 ± 1.88e − 01**	2.05e + 01 ± 1.28e − 01	+
f_9	**6.09e + 00 ± 3.38e + 00**	1.17e + 01 ± 4.19e + 00	+
f_{10}	**1.49e − 02 ± 1.24e − 02**	1.68e − 02 ± 1.90e − 02	=
f_{11}	**4.17e + 00 ± 1.67e + 00**	5.39e + 01 ± 1.21e + 02	+
f_{12}	**1.44e + 01 ± 6.82e + 00**	1.19e + 02 ± 2.23e + 02	+
f_{13}	**8.02e + 01 ± 9.29e + 01**	1.35e + 02 ± 2.35e + 02	=
f_{14}	**1.33e + 02 ± 9.00e + 01**	1.28e + 03 ± 7.00e + 02	+
f_{15}	**7.14e + 02 ± 1.99e + 02**	1.63e + 03 ± 4.24e + 02	+
f_{16}	**3.19e − 01 ± 2.33e − 01**	4.24e − 01 ± 3.88e − 01	=
f_{17}	**1.12e + 01 ± 3.79e + 00**	4.22e + 02 ± 5.16e + 02	+
f_{18}	**6.83e + 01 ± 8.47e + 01**	4.28e + 02 ± 4.65e + 02	+
f_{19}	**9.05e − 01 ± 3.26e − 01**	1.03e + 00 ± 4.25e − 01	=
f_{20}	**3.92e + 00 ± 4.48e − 01**	4.34e + 00 ± 5.07e − 01	+
f_{21}	**2.43e + 02 ± 1.19e + 02**	3.62e + 02 ± 8.81e + 01	+
f_{22}	**2.70e + 02 ± 2.26e + 02**	1.52e + 03 ± 8.56e + 02	+
f_{23}	**9.97e + 02 ± 3.34e + 02**	2.24e + 03 ± 4.17e + 02	+
f_{24}	**1.37e + 02 ± 3.81e + 01**	3.00e + 02 ± 1.16e + 02	+
f_{25}	**1.99e + 02 ± 3.06e + 01**	2.43e + 02 ± 5.09e + 01	+
f_{26}	**1.48e + 02 ± 4.29e + 01**	2.45e + 02 ± 1.12e + 02	+
f_{27}	**3.56e + 02 ± 8.99e + 01**	4.06e + 02 ± 1.20e + 02	+
f_{28}	**2.38e + 02 ± 9.25e + 01**	1.03e + 03 ± 1.08e + 03	+

Table 3
SPAM against SPAM$^{0.5}$ on CEC2005 in 30D.

	SPAM	SPAM$^{0.5}$	
f_1	**0.00e + 00 ± 4.96e − 14**	0.00e + 00 ± 5.18e − 14	=
f_2	**0.00e + 00 ± 5.45e − 14**	5.68e − 14 ± 1.61e − 14	=
f_3	1.87e + 03 ± 1.39e + 03	**1.75e + 03 ± 1.64e + 03**	=
f_4	**2.47e + 04 ± 9.22e + 03**	1.87e + 05 ± 4.34e + 05	+
f_5	**8.21e + 02 ± 4.69e + 02**	8.26e + 02 ± 4.37e + 02	=
f_6	**4.69e + 01 ± 9.38e + 01**	4.73e + 01 ± 8.79e + 01	+
f_7	**6.41e − 04 ± 2.19e − 03**	4.80e + 03 ± 1.21e + 02	+
f_8	**2.00e + 01 ± 8.76e − 04**	2.03e + 01 ± 3.63e − 01	+
f_9	**2.13e − 10 ± 5.06e − 11**	1.66e + 02 ± 1.93e + 02	+
f_{10}	**6.83e + 01 ± 3.22e + 01**	2.09e + 02 ± 3.08e + 02	=
f_{11}	**1.35e + 01 ± 5.15e + 00**	1.59e + 01 ± 9.74e + 00	=
f_{12}	**4.29e + 02 ± 7.94e + 02**	1.24e + 03 ± 2.04e + 03	+
f_{13}	**2.79e + 00 ± 6.01e − 01**	3.24e + 00 ± 7.94e − 01	+
f_{14}	**4.39e + 01 ± 2.75e − 01**	4.43e + 01 ± 4.60e − 01	+
f_{15}	**1.28e + 02 ± 1.13e + 02**	4.26e + 02 ± 2.54e + 02	+
f_{16}	**1.36e + 02 ± 5.52e + 01**	3.14e + 02 ± 2.87e + 02	+
f_{17}	**2.22e + 02 ± 5.44e + 01**	5.36e + 02 ± 2.79e + 02	+
f_{18}	**9.03e + 02 ± 3.04e + 01**	9.38e + 02 ± 7.71e + 01	+
f_{19}	**8.95e + 02 ± 4.00e + 01**	9.38e + 02 ± 8.13e + 01	+
f_{20}	**9.00e + 02 ± 5.68e + 01**	9.45e + 02 ± 1.17e + 02	+
f_{21}	**4.99e + 02 ± 9.02e + 00**	5.77e + 02 ± 1.89e + 02	=
f_{22}	9.04e + 02 ± 2.92e + 01	**9.00e + 02 ± 2.83e + 01**	=
f_{23}	**5.32e + 02 ± 2.00e + 01**	6.45e + 02 ± 2.11e + 02	+
f_{24}	**2.00e + 02 ± 3.80e − 11**	2.21e + 02 ± 1.17e + 02	=
f_{25}	**1.24e + 03 ± 4.43e + 02**	1.67e + 03 ± 1.24e + 01	+

Table 4
SPAM against SPAM$^{0.5}$ on CEC2013 in 50D.

	SPAM	SPAM$^{0.5}$	
f_1	2.27e − 13 ± 2.27e − 14	**2.27e − 13 ± 0.00e + 00**	=
f_2	3.76e + 04 ± 1.97e + 04	**3.35e + 04 ± 1.63e + 04**	=
f_3	**7.86e + 06 ± 1.32e + 07**	9.00e + 06 ± 1.51e + 07	=
f_4	1.20e + 03 ± 4.28e + 03	**5.84e + 02 ± 4.99e + 02**	=
f_5	**4.29e − 07 ± 5.98e − 08**	2.11e − 05 ± 2.10e − 05	+
f_6	**2.61e + 01 ± 1.90e + 01**	4.14e + 01 ± 9.74e + 00	+
f_7	**4.78e + 01 ± 2.15e + 01**	3.20e + 04 ± 3.18e + 05	=
f_8	**2.11e + 01 ± 6.29e − 02**	2.12e + 01 ± 4.92e − 02	+
f_9	**4.77e + 01 ± 3.86e + 00**	7.23e + 01 ± 1.22e + 01	+
f_{10}	**1.08e − 02 ± 7.37e − 03**	2.42e − 02 ± 1.64e − 02	+
f_{11}	**5.03e + 01 ± 1.03e + 01**	1.76e + 02 ± 4.36e + 02	+
f_{12}	**3.17e + 02 ± 1.21e + 02**	1.32e + 03 ± 1.39e + 03	+
f_{13}	**4.88e + 02 ± 8.32e + 01**	1.98e + 03 ± 1.65e + 03	+
f_{14}	**1.28e + 03 ± 2.81e + 02**	5.11e + 03 ± 3.24e + 03	+
f_{15}	**6.48e + 03 ± 5.37e + 02**	8.86e + 03 ± 1.08e + 03	+
f_{16}	8.59e − 02 ± 3.78e − 02	**8.53e − 02 ± 5.09e − 02**	=
f_{17}	**9.42e + 01 ± 9.78e + 00**	3.97e + 03 ± 3.34e + 03	+
f_{18}	**5.08e + 02 ± 8.68e + 01**	4.06e + 03 ± 3.21e + 03	+
f_{19}	**5.05e + 00 ± 9.04e − 01**	5.62e + 00 ± 1.32e + 00	+
f_{20}	**2.43e + 01 ± 5.55e − 01**	2.47e + 01 ± 4.44e − 01	+
f_{21}	**4.37e + 02 ± 3.36e + 02**	7.58e + 02 ± 3.69e + 02	+
f_{22}	**1.93e + 03 ± 4.31e + 02**	8.10e + 03 ± 4.44e + 03	+
f_{23}	**8.78e + 03 ± 1.26e + 03**	1.17e + 04 ± 1.13e + 03	+
f_{24}	**3.25e + 02 ± 2.43e + 01**	1.47e + 03 ± 1.06e + 03	+
f_{25}	**3.68e + 02 ± 1.34e + 01**	4.68e + 02 ± 1.72e + 02	+
f_{26}	**2.24e + 02 ± 6.92e + 01**	6.05e + 02 ± 7.20e + 02	+
f_{27}	**1.28e + 03 ± 1.97e + 02**	1.29e + 03 ± 3.51e + 02	=
f_{28}	**1.42e + 03 ± 1.96e + 03**	3.73e + 03 ± 5.80e + 03	=

- Modified Differential Evolution with p-Best Crossover (MDE-pBX) proposed in [78] with population size equal to 100 individuals and group size q equal to 15% of the population size.
- Cooperatively Coevolving Particle Swarms Optimizer (CCPSO2) proposed in [79] with population size equal to 30 individuals, Cauchy/Gaussian-sampling selection probability $p = 0.5$ and set of potential group sizes $S = \{2, 5, 10\}, S = \{2, 5, 10, 50, 100\}, S = \{2, 5, 10, 50, 100, 250\}$ for experiments in 30, 100 and 1000 dimensions, respectively.

Table 5
SPAM against SPAM$^{0.5}$ on BBOB2010 in 100D.

	SPAM	SPAM$^{0.5}$	
f_1	$\mathbf{2.42e-13 \pm 2.13e-13}$	$2.42e-13 \pm 2.13e-13$	=
f_2	$\mathbf{3.21e-07 \pm 7.33e-08}$	$2.82e+02 \pm 3.48e+02$	=
f_3	$\mathbf{9.60e+01 \pm 1.24e+01}$	$1.84e+02 \pm 8.30e+01$	+
f_4	$\mathbf{1.28e+02 \pm 1.86e+01}$	$2.79e+02 \pm 1.30e+02$	+
f_5	$\mathbf{1.31e-04 \pm 1.27e-05}$	$1.18e+02 \pm 1.05e+02$	=
f_6	$\mathbf{3.93e-08 \pm 3.87e-08}$	$5.05e-08 \pm 6.71e-08$	=
f_7	$\mathbf{5.07e+01 \pm 1.35e+01}$	$5.45e+01 \pm 1.40e+01$	+
f_8	$\mathbf{3.45e+01 \pm 1.31e+01}$	$5.41e+01 \pm 2.28e+01$	+
f_9	$\mathbf{4.70e+01 \pm 1.14e+01}$	$5.87e+01 \pm 1.80e+01$	+
f_{10}	$6.98e+02 \pm 2.21e+02$	$\mathbf{6.11e+02 \pm 2.04e+02}$	−
f_{11}	$\mathbf{8.15e+01 \pm 2.76e+01}$	$1.01e+02 \pm 3.72e+01$	+
f_{12}	$\mathbf{1.97e-02 \pm 6.48e-02}$	$2.01e-02 \pm 8.77e-02$	=
f_{13}	$\mathbf{7.74e-01 \pm 9.16e-01}$	$1.63e+00 \pm 3.10e+00$	=
f_{14}	$4.21e-05 \pm 8.74e-06$	$\mathbf{3.99e-05 \pm 8.28e-06}$	=
f_{15}	$\mathbf{2.68e+02 \pm 3.86e+01}$	$2.78e+02 \pm 4.35e+01$	=
f_{16}	$\mathbf{2.35e+00 \pm 8.17e-01}$	$2.37e+00 \pm 9.12e-01$	=
f_{17}	$\mathbf{7.93e+00 \pm 3.86e+00}$	$1.01e+01 \pm 4.78e+00$	+
f_{18}	$\mathbf{1.65e+01 \pm 9.86e+00}$	$1.91e+01 \pm 1.36e+01$	=
f_{19}	$\mathbf{1.66e+00 \pm 3.11e-01}$	$2.18e+00 \pm 1.20e+00$	+
f_{20}	$\mathbf{1.27e+00 \pm 1.47e-01}$	$1.65e+00 \pm 2.29e-01$	+
f_{21}	$\mathbf{3.42e+00 \pm 3.50e+00}$	$1.23e+01 \pm 1.27e+01$	+
f_{22}	$\mathbf{6.27e+00 \pm 7.42e+00}$	$1.56e+01 \pm 1.18e+01$	+
f_{23}	$\mathbf{6.62e-01 \pm 2.75e-01}$	$1.42e+00 \pm 8.45e-01$	+
f_{24}	$\mathbf{3.11e+02 \pm 5.78e+01}$	$3.24e+02 \pm 7.46e+01$	=

Table 6
SPAM against SPAM$^{0.5}$ on CEC2008 in 1000D.

	SPAM	SPAM$^{0.5}$	
f_1	$5.12e-13 \pm 6.32e-13$	$\mathbf{5.12e-13 \pm 6.31e-13}$	=
f_2	$\mathbf{1.32e-01 \pm 3.27e-02}$	$1.17e+01 \pm 1.17e+01$	+
f_3	$\mathbf{8.97e+02 \pm 2.83e+01}$	$9.28e+02 \pm 4.42e+01$	+
f_4	$\mathbf{7.17e+01 \pm 1.16e+01}$	$5.39e+03 \pm 5.28e+03$	+
f_5	$\mathbf{7.56e-05 \pm 7.36e-04}$	$6.65e-04 \pm 2.51e-03$	=
f_6	$\mathbf{1.22e+01 \pm 9.51e+00}$	$1.73e+01 \pm 6.53e+00$	+

Table 7
SPAM against SPAM$^{0.5}$ on CEC2010 in 1000D.

	SPAM	SPAM$^{0.5}$	
f_1	$\mathbf{1.92e-06 \pm 1.33e-07}$	$3.12e+06 \pm 3.07e+06$	=
f_2	$\mathbf{6.15e+01 \pm 1.28e+01}$	$4.58e+03 \pm 4.95e+03$	+
f_3	$\mathbf{8.06e+00 \pm 9.49e+00}$	$1.75e+01 \pm 6.33e+00$	+
f_4	$\mathbf{1.14e+11 \pm 4.32e+10}$	$2.89e+11 \pm 2.00e+11$	+
f_5	$6.78e+08 \pm 9.14e+07$	$\mathbf{6.64e+08 \pm 1.01e+08}$	=
f_6	$\mathbf{1.98e+07 \pm 7.24e+04}$	$1.98e+07 \pm 6.30e+04$	=
f_7	$\mathbf{4.17e+05 \pm 5.05e+05}$	$3.06e+06 \pm 3.04e+06$	+
f_8	$\mathbf{3.17e+06 \pm 2.82e+06}$	$9.62e+06 \pm 1.24e+07$	+
f_9	$\mathbf{3.21e+06 \pm 3.48e+05}$	$5.13e+06 \pm 2.18e+06$	+
f_{10}	$\mathbf{5.37e+03 \pm 3.84e+02}$	$7.80e+03 \pm 2.46e+03$	+
f_{11}	$\mathbf{2.00e+02 \pm 2.22e+01}$	$2.12e+02 \pm 1.83e+01$	+
f_{12}	$\mathbf{3.76e-03 \pm 1.40e-03}$	$8.17e-02 \pm 8.90e-02$	+
f_{13}	$\mathbf{1.22e+03 \pm 8.23e+02}$	$1.42e+03 \pm 8.84e+02$	=
f_{14}	$\mathbf{4.65e+06 \pm 4.18e+05}$	$7.20e+06 \pm 2.10e+06$	+
f_{15}	$\mathbf{1.00e+04 \pm 1.04e+03}$	$1.03e+04 \pm 6.44e+02$	=
f_{16}	$\mathbf{2.64e+02 \pm 1.26e+02}$	$3.70e+02 \pm 7.64e+01$	+
f_{17}	$\mathbf{4.36e-01 \pm 1.13e-01}$	$1.03e+01 \pm 1.19e+01$	+
f_{18}	$\mathbf{1.71e+03 \pm 1.09e+03}$	$2.19e+03 \pm 1.05e+03$	+
f_{19}	$\mathbf{3.32e+05 \pm 3.25e+04}$	$5.56e+05 \pm 2.27e+05$	+
f_{20}	$\mathbf{8.92e+02 \pm 3.28e+01}$	$9.31e+02 \pm 4.52e+01$	+

- Memetic Algorithm with CMA-ES Chains (MACh) proposed in [34] with population size equal to 60 individuals, probability of updating a chromosome by mutation equal to 0.125, local/global search ratio $r_{\frac{L}{G}} = 0.5$, BLX-α crossover with $\alpha = 0.5, n_{ass}$ parameter for Negative Assortative Mating set to 3, LS intensity stretch $I_{str} = 500$ and threshold $\delta_{LS}^{min} = 10^{-8}$. In 1000 dimensions the version employing Solis-Wetz algorithm has been used as suggested in [80].

It must be remarked that these three algorithms are based on diverse search logics, have been recently proposed and are amongst the optimization algorithms that display the best performance. Numerical results are shown in Tables 8–13.

It can be easily observed that SPAM displays the best performance with respect to the other three modern algorithms under consideration as it significantly outperforms MDE-pBX, CCPSO2, and MA-CMA, in respectively 88, 94 and, 81 cases out of the 132 problems studied in this paper. It must be observed that while the other state-of-the-art tend to be specialized to solve some classes of problems, SPAM appears to efficiently tackle a large range of problems. For example, while MACh displays a very good performance in low dimensions, it pays off with a not so good performance in large scale problems. On the contrary, SPAM, thanks to its problem analyser, coordinates its component in a versatile way, thus offering a robust performance.

In order to further enhance the statistical significance of the numerical results the Holm–Bonferroni procedure, see [81,82], has been applied for the 4 algorithms under study and the 132 problems under consideration. This statistical test confirms the superiority of SPAM with respect to the other algorithms and problems taken into account in this paper. Numerical results are summarized in Table 14.

4.3. Computational cost of the separability analysis

In order to make an evaluation of the computational cost of the analysis on the separability, its calculation time has been saved 100 times for a,n easily scalable problem (a sphere function) in 10, 30, 50, 100, and 1000. The average computational time has been calculated over the 100 runs available for each dimensionality value. The cost of the fitness evaluation has then been estimated and removed from the computational time. A diagram showing the computational cost for the considered dimensionality values is shown in Fig. 9.

It can be observed that the computational cost of this analysis grows quadratically with the dimensionality as it requires the use of a covariance matrix. More specifically, the analysis on separability has approximately the same computational cost of the CMAES (for 20% of the run) as the other operations (e.g. calculation of the Pearson coefficient) are characterized by a computational cost negligible with respect to that of the covariance matrix update. As a final remark, the separability analysis does not only characterize the separability of the problem but also optimizes the function as improved solutions are usable by the second part of the optimization.

Table 8
SPAM against MDE-pBX, CCPSO2, and MACh on CEC2013 in 10D.

	SPAM	MDE-pBX		CCPSO2		MACh	
f_1	$\mathbf{0.00e+00 \pm 0.00e+00}$	$0.00e+00 \pm 0.00e+00$	=	$3.37e-03 \pm 2.16e-02$	+	$0.00e+00 \pm 0.00e+00$	=
f_2	$\mathbf{0.00e+00 \pm 0.00e+00}$	$2.06e+03 \pm 4.90e+03$	+	$1.61e+06 \pm 1.21e+06$	+	$4.39e+02 \pm 4.16e+03$	+
f_3	$\mathbf{4.45e+01 \pm 2.61e+02}$	$9.43e+04 \pm 4.32e+05$	+	$6.19e+07 \pm 1.02e+08$	+	$3.19e+04 \pm 1.94e+05$	=
f_4	$\mathbf{0.00e+00 \pm 0.00e+00}$	$1.38e+00 \pm 7.24e+00$	+	$1.05e+04 \pm 2.93e+03$	+	$6.06e+01 \pm 4.97e+02$	+
f_5	$2.36e-08 \pm 2.77e-08$	$\mathbf{0.00e+00 \pm 8.73e-14}$	−	$1.25e-02 \pm 2.41e-02$	+	$8.59e-09 \pm 4.44e-08$	−
f_6	$3.94e+00 \pm 4.67e+00$	$5.71e+00 \pm 4.83e+00$	=	$4.56e+00 \pm 4.46e+00$	+	$\mathbf{5.14e-02 \pm 4.11e-01}$	−
f_7	$6.67e+01 \pm 5.65e+01$	$7.99e+00 \pm 1.04e+01$	−	$4.16e+01 \pm 1.25e+01$	−	$\mathbf{1.86e+00 \pm 3.52e+00}$	−
f_8	$\mathbf{2.03e+01 \pm 1.88e-01}$	$2.05e+01 \pm 1.07e-01$	+	$2.04e+01 \pm 8.82e-02$	+	$2.04e+01 \pm 1.01e-01$	+
f_9	$6.09e+00 \pm 3.38e+00$	$2.11e+00 \pm 1.47e+00$	−	$5.58e+00 \pm 1.01e+00$	−	$\mathbf{1.78e+00 \pm 9.45e-01}$	−
f_{10}	$\mathbf{1.49e-02 \pm 1.24e-02}$	$1.16e-01 \pm 9.01e-02$	+	$2.08e+00 \pm 1.15e+00$	+	$1.47e-01 \pm 1.68e-01$	+
f_{11}	$4.17e+00 \pm 1.67e+00$	$2.86e+00 \pm 1.86e+00$	−	$3.44e+00 \pm 2.02e+00$	−	$\mathbf{1.96e+00 \pm 1.16e+00}$	−
f_{12}	$1.44e+01 \pm 6.82e+00$	$1.13e+01 \pm 4.99e+00$	−	$3.18e+01 \pm 9.05e+00$	+	$\mathbf{5.25e+00 \pm 3.31e+00}$	−
f_{13}	$8.02e+01 \pm 9.29e+01$	$1.89e+01 \pm 9.18e+00$	−	$4.05e+01 \pm 8.05e+00$	=	$\mathbf{7.75e+00 \pm 4.15e+00}$	−
f_{14}	$1.33e+02 \pm 9.00e+01$	$1.09e+02 \pm 1.07e+02$	−	$8.02e+01 \pm 5.47e+01$	−	$\mathbf{4.22e+01 \pm 6.04e+01}$	−
f_{15}	$7.14e+02 \pm 1.99e+02$	$7.42e+02 \pm 2.77e+02$	=	$1.02e+03 \pm 2.89e+02$	+	$\mathbf{5.78e+02 \pm 2.04e+02}$	−
f_{16}	$\mathbf{3.19e-01 \pm 2.33e-01}$	$5.80e-01 \pm 4.09e-01$	+	$1.31e+00 \pm 2.81e-01$	+	$4.88e+00 \pm 1.13e+00$	+
f_{17}	$\mathbf{1.12e+01 \pm 3.79e+00}$	$1.32e+01 \pm 1.82e+00$	+	$1.74e+01 \pm 2.48e+00$	+	$2.62e+02 \pm 3.30e+01$	+
f_{18}	$6.83e+01 \pm 8.47e+01$	$\mathbf{2.01e+01 \pm 5.56e+00}$	−	$5.92e+01 \pm 8.00e+00$	−	$3.43e+02 \pm 4.64e+01$	+
f_{19}	$9.05e-01 \pm 3.26e-01$	$\mathbf{6.09e-01 \pm 2.13e-01}$	−	$1.01e+00 \pm 3.27e-01$	+	$2.76e+02 \pm 1.37e+02$	+
f_{20}	$3.92e+00 \pm 4.48e-01$	$\mathbf{2.87e+00 \pm 6.22e-01}$	−	$3.60e+00 \pm 2.21e-01$	−	$4.99e+00 \pm 5.33e-02$	+
f_{21}	$\mathbf{2.43e+02 \pm 1.19e+02}$	$3.98e+02 \pm 1.99e+01$	+	$3.88e+02 \pm 3.73e+01$	+	$6.36e+02 \pm 4.30e+01$	+
f_{22}	$2.70e+02 \pm 2.26e+02$	$1.79e+02 \pm 1.41e+02$	−	$\mathbf{1.13e+02 \pm 6.86e+01}$	−	$7.07e+02 \pm 2.14e+02$	+
f_{23}	$9.97e+02 \pm 3.34e+02$	$9.21e+02 \pm 3.68e+02$	=	$1.31e+03 \pm 2.94e+02$	+	$\mathbf{8.83e+02 \pm 9.94e+01}$	−
f_{24}	$\mathbf{1.37e+02 \pm 3.81e+01}$	$2.03e+02 \pm 1.05e+01$	+	$2.08e+02 \pm 1.99e+01$	+	$2.56e+02 \pm 1.08e+01$	+
f_{25}	$\mathbf{1.99e+02 \pm 3.06e+01}$	$2.01e+02 \pm 8.76e+00$	+	$2.15e+02 \pm 8.90e+00$	+	$2.48e+02 \pm 5.43e+00$	+
f_{26}	$\mathbf{1.48e+02 \pm 4.29e+01}$	$1.55e+02 \pm 4.37e+01$	=	$1.69e+02 \pm 2.62e+01$	+	$3.35e+02 \pm 4.65e+01$	+
f_{27}	$3.56e+02 \pm 8.99e+01$	$\mathbf{3.13e+02 \pm 4.68e+01}$	−	$4.46e+02 \pm 6.43e+01$	+	$1.03e+03 \pm 1.27e+02$	+
f_{28}	$\mathbf{2.38e+02 \pm 9.25e+01}$	$3.01e+02 \pm 6.82e+01$	+	$3.87e+02 \pm 1.80e+02$	+	$1.30e+03 \pm 7.99e+01$	+

Table 9
SPAM against MDE-pBX, CCPSO2, and MACh on CEC2005 in 30D.

	SPAM	MDE-pBX		CCPSO2		MACh	
f_1	**0.00e+00 ± 4.96e−14**	5.68e−14 ± 4.13e−13	+	1.71e−13 ± 4.62e−13	+	8.90e−09 ± 8.36e−10	+
f_2	**0.00e+00 ± 5.45e−14**	9.75e−04 ± 4.18e−03	+	9.47e+00 ± 1.66e+01	+	1.57e−03 ± 1.47e−02	+
f_3	**1.87e+03 ± 1.39e+03**	3.23e+05 ± 2.14e+05	+	1.53e+06 ± 6.74e+05	+	1.46e+05 ± 4.32e+05	+
f_4	2.47e+04 ± 9.22e+03	**2.31e+02 ± 6.07e+02**	−	1.80e+04 ± 4.47e+03	−	7.71e+02 ± 5.06e+02	−
f_5	**8.21e+02 ± 4.69e+02**	2.99e+03 ± 5.85e+02	+	9.44e+03 ± 1.72e+03	+	8.70e+02 ± 5.63e+02	=
f_6	4.69e+01 ± 9.38e+01	3.38e+01 ± 3.37e+01	−	6.99e+01 ± 5.60e+01	+	**9.54e+00 ± 3.07e+01**	−
f_7	**6.41e−04 ± 2.19e−03**	3.16e+04 ± 2.04e+05	+	2.49e−02 ± 2.10e−02	+	3.86e+01 ± 1.16e+02	+
f_8	**2.00e+01 ± 8.76e−04**	2.07e+01 ± 4.12e−01	+	2.10e+01 ± 6.31e−02	+	2.00e+01 ± 9.08e−03	+
f_9	**2.13e−10 ± 5.06e−11**	4.02e+01 ± 1.09e+01	+	5.67e−01 ± 6.74e−01	+	2.09e−01 ± 7.35e−01	+
f_{10}	6.83e+01 ± 3.22e+01	7.31e+01 ± 1.88e+01	+	2.01e+02 ± 5.34e+01	+	**3.36e+01 ± 2.17e+01**	−
f_{11}	**1.35e+01 ± 5.15e+00**	2.10e+01 ± 4.91e+00	+	2.80e+01 ± 2.36e+00	+	2.36e+01 ± 3.18e+00	+
f_{12}	**4.29e+02 ± 7.94e+02**	3.95e+03 ± 3.90e+03	+	3.51e+03 ± 3.78e+03	+	7.06e+02 ± 1.07e+03	=
f_{13}	2.79e+00 ± 6.01e−01	3.49e+00 ± 1.16e+00	+	**8.97e−01 ± 1.83e−01**	−	3.07e+00 ± 1.96e+00	=
f_{14}	4.39e+01 ± 2.75e−01	**4.28e+01 ± 4.14e−01**	−	4.30e+01 ± 2.93e−01	−	4.29e+01 ± 3.18e−01	−
f_{15}	**1.28e+02 ± 1.13e+02**	3.55e+02 ± 7.82e+01	+	2.44e+02 ± 1.45e+02	+	3.07e+02 ± 2.92e+01	+
f_{16}	**1.36e+02 ± 5.52e+01**	1.79e+02 ± 1.40e+02	=	2.88e+02 ± 1.05e+02	+	1.41e+02 ± 1.62e+02	+
f_{17}	2.22e+02 ± 5.44e+01	**1.81e+02 ± 1.38e+02**	−	3.77e+02 ± 7.25e+01	+	1.85e+02 ± 1.68e+02	−
f_{18}	9.03e+02 ± 3.04e+01	**8.99e+02 ± 5.98e+01**	−	9.30e+02 ± 1.31e+01	+	9.01e+02 ± 3.88e+01	−
f_{19}	**8.95e+02 ± 4.00e+01**	8.99e+02 ± 5.81e+01	+	9.28e+02 ± 1.41e+01	+	8.97e+02 ± 4.09e+01	+
f_{20}	9.00e+02 ± 5.68e+01	**8.97e+02 ± 5.96e+01**	−	9.24e+02 ± 2.88e+01	+	8.99e+02 ± 3.94e+01	−
f_{21}	**4.99e+02 ± 9.02e+00**	5.63e+02 ± 1.91e+02	+	5.04e+02 ± 3.07e+01	+	5.22e+02 ± 1.13e+02	+
f_{22}	9.04e+02 ± 2.92e+01	9.74e+02 ± 2.88e+01	+	1.06e+03 ± 2.97e+01	+	**8.89e+02 ± 1.31e+01**	−
f_{23}	**5.32e+02 ± 2.00e+01**	5.91e+02 ± 1.48e+02	+	5.32e+02 ± 2.00e+01	+	5.40e+02 ± 6.27e+01	=
f_{24}	2.00e+02 ± 3.80e−11	2.10e+02 ± 9.81e+01	+	2.00e+02 ± 1.67e−06	+	**2.00e+02 ± 0.00e+00**	−
f_{25}	**1.24e+03 ± 4.43e+02**	1.59e+03 ± 1.32e+02	+	1.59e+03 ± 6.89e+01	+	1.63e+03 ± 8.19e+00	+

Table 10
SPAM against MDE-pBX, CCPSO2, MACh, on CEC2013 in 50D.

	SPAM	MDE-pBX		CCPSO2		MACh	
f_1	2.27e−13 ± 2.27e−14	5.58e−10 ± 4.68e−09	+	1.36e−12 ± 3.64e−12	+	**0.00e+00 ± 0.00e+00**	−
f_2	**3.76e+04 ± 1.97e+04**	8.12e+05 ± 4.07e+05	+	4.43e+06 ± 2.70e+06	+	4.14e+05 ± 4.67e+05	+
f_3	7.86e+06 ± 1.32e+07	1.63e+08 ± 2.00e+08	+	3.47e+09 ± 3.70e+09	+	**3.91e+06 ± 1.07e+07**	−
f_4	1.20e+03 ± 4.28e+03	**9.70e+02 ± 7.38e+02**	−	1.08e+05 ± 3.85e+04	+	2.85e+04 ± 8.23e+03	+
f_5	**4.29e−07 ± 5.98e−08**	2.11e−06 ± 1.97e−05	+	5.73e−05 ± 5.68e−04	+	1.19e−06 ± 4.52e−06	+
f_6	**2.61e+01 ± 1.90e+01**	5.81e+01 ± 2.32e+01	+	4.86e+01 ± 1.64e+01	+	3.77e+01 ± 1.14e+01	+
f_7	4.78e+01 ± 2.15e+01	6.94e+01 ± 1.27e+01	+	1.44e+02 ± 2.81e+01	+	**1.42e+01 ± 5.65e+00**	−
f_8	**2.11e+01 ± 6.29e−02**	2.12e+01 ± 4.66e−02	+	2.12e+01 ± 3.78e−02	+	2.12e+01 ± 3.48e−02	+
f_9	4.77e+01 ± 3.86e+00	4.42e+01 ± 8.00e+00	−	5.88e+01 ± 3.72e+00	+	**2.48e+01 ± 1.27e+01**	−
f_{10}	**1.08e−02 ± 7.37e−03**	4.03e−01 ± 5.14e−01	+	2.08e−01 ± 2.82e−01	+	1.58e−01 ± 1.41e−01	+
f_{11}	5.03e+01 ± 1.03e+01	1.23e+02 ± 2.63e+01	+	**9.48e−01 ± 8.55e−01**	−	1.86e+01 ± 6.22e+00	−
f_{12}	3.17e+02 ± 1.21e+02	1.59e+02 ± 3.72e+01	−	4.63e+02 ± 9.91e+01	+	**1.39e+02 ± 8.05e+01**	−
f_{13}	4.88e+02 ± 8.32e+01	3.18e+02 ± 5.58e+01	−	5.70e+02 ± 8.52e+01	+	**1.83e+02 ± 5.41e+01**	−
f_{14}	1.28e+03 ± 2.81e+02	2.74e+03 ± 8.02e+02	+	**6.98e+00 ± 3.64e+00**	−	1.97e+03 ± 2.41e+03	+
f_{15}	**6.48e+03 ± 5.37e+02**	7.58e+03 ± 8.91e+02	+	8.25e+03 ± 8.01e+02	+	7.62e+03 ± 9.47e+02	+
f_{16}	**8.59e−02 ± 3.78e−02**	1.85e+00 ± 8.03e−01	+	2.71e+00 ± 5.64e−01	+	7.43e+00 ± 9.69e−01	+
f_{17}	9.42e+01 ± 9.78e+00	1.76e+02 ± 3.52e+01	+	**5.17e+01 ± 4.23e−01**	−	2.79e+02 ± 7.07e+01	+
f_{18}	5.08e+02 ± 8.68e+01	**1.94e+02 ± 3.77e+01**	−	5.08e+02 ± 1.03e+02	=	3.95e+02 ± 5.05e+00	−
f_{19}	5.05e+00 ± 9.04e−01	4.27e+01 ± 2.78e+01	+	**1.46e+00 ± 2.30e−01**	−	4.71e+02 ± 5.91e+01	+
f_{20}	2.43e+01 ± 5.55e−01	**2.00e+01 ± 1.03e+00**	−	2.32e+01 ± 7.70e−01	−	2.50e+01 ± 1.34e−05	+
f_{21}	4.37e+02 ± 3.36e+02	8.54e+02 ± 3.66e+02	+	**4.06e+02 ± 3.22e+02**	=	9.64e+02 ± 1.81e+02	+
f_{22}	1.93e+03 ± 4.31e+02	3.09e+03 ± 9.77e+02	+	**9.40e+01 ± 9.52e+01**	−	7.62e+02 ± 2.05e+02	−
f_{23}	**8.78e+03 ± 1.26e+03**	9.03e+03 ± 1.29e+03	=	1.07e+04 ± 1.17e+03	+	8.88e+03 ± 1.67e+03	=
f_{24}	3.25e+02 ± 2.43e+01	**2.85e+02 ± 1.44e+01**	−	3.61e+02 ± 1.04e+01	+	8.62e+02 ± 1.08e+02	+
f_{25}	**3.68e+02 ± 1.34e+01**	3.68e+02 ± 1.45e+01	=	4.00e+02 ± 1.07e+01	+	6.79e+02 ± 3.82e+01	+
f_{26}	2.24e+02 ± 6.92e+01	3.48e+02 ± 8.03e+01	+	**2.22e+02 ± 6.42e+01**	−	5.98e+02 ± 7.45e+01	+
f_{27}	1.28e+03 ± 1.97e+02	1.24e+03 ± 1.37e+02	=	1.82e+03 ± 1.67e+02	+	**1.18e+03 ± 3.70e+02**	=
f_{28}	1.42e+03 ± 1.96e+03	7.52e+02 ± 1.06e+03	−	**5.77e+02 ± 7.84e+02**	−	1.40e+03 ± 3.98e+02	−

4.4. Extended numerical results

In addition to the algorithms mentioned above, SPAM has been compared with the following modern algorithms:

- Comprehensive Learning Particle Swarm Optimizer (CLPSO) proposed in [83] with population size equal to 60 individuals.

Table 11
SPAM against MDE-pBX, CCPSO2, and MACh on BBOB2010 in 100D.

	SPAM	MDE-pBX		CCPSO2		MACh	
f_1	**2.42e − 13 ± 2.13e − 13**	7.73e − 06 ± 7.60e − 05	+	4.12e − 13 ± 1.97e − 13	+	9.30e − 09 ± 6.55e − 10	+
f_2	3.21e − 07 ± 7.33e − 08	1.11e − 03 ± 6.06e − 03	+	**9.66e − 13 ± 1.77e − 12**	−	1.44e − 06 ± 5.31e − 06	+
f_3	9.60e + 01 ± 1.24e + 01	4.95e + 02 ± 7.94e + 01	+	**8.09e + 00 ± 8.40e + 00**	−	4.86e + 01 ± 9.03e + 00	−
f_4	1.28e + 02 ± 1.86e + 01	8.65e + 02 ± 1.31e + 02	+	**2.24e + 01 ± 1.31e + 01**	−	8.02e + 01 ± 1.14e + 01	−
f_5	1.31e − 04 ± 1.27e − 05	9.18e + 00 ± 1.27e + 01	+	2.41e − 04 ± 1.20e − 03	+	**1.90e − 13 ± 0.00e + 00**	−
f_6	**3.93e − 08 ± 3.87e − 08**	4.44e + 01 ± 3.32e + 01	+	8.94e + 01 ± 4.02e + 01	+	3.25e − 04 ± 1.60e − 03	=
f_7	5.07e + 01 ± 1.35e + 01	2.77e + 02 ± 7.43e + 01	+	3.45e + 02 ± 4.90e + 01	+	**4.04e + 01 ± 7.77e + 00**	−
f_8	3.45e + 01 ± 1.31e + 01	1.91e + 02 ± 6.77e + 01	+	1.20e + 02 ± 3.42e + 01	+	**9.41e + 00 ± 1.45e + 01**	−
f_9	**4.70e + 01 ± 1.14e + 01**	1.28e + 02 ± 3.75e + 01	+	1.06e + 02 ± 2.78e + 01	+	5.57e + 01 ± 8.49e + 00	+
f_{10}	**6.98e + 02 ± 2.21e + 02**	1.65e + 04 ± 7.99e + 03	+	2.62e + 04 ± 6.64e + 03	+	2.45e + 03 ± 1.44e + 03	+
f_{11}	8.15e + 01 ± 2.76e + 01	**1.54e + 01 ± 7.45e + 00**	−	5.45e + 02 ± 1.95e + 02	+	1.09e + 02 ± 1.85e + 01	+
f_{12}	**1.97e − 02 ± 6.48e − 02**	2.21e + 01 ± 7.07e + 01	+	7.95e + 00 ± 1.20e + 01	+	1.43e − 01 ± 6.95e − 01	=
f_{13}	**7.74e − 01 ± 9.16e − 01**	4.76e + 00 ± 6.70e + 00	+	3.16e + 00 ± 4.20e + 00	+	1.12e + 00 ± 1.76e + 00	=
f_{14}	4.21e − 05 ± 8.74e − 06	2.73e − 03 ± 2.55e − 03	+	1.32e − 03 ± 2.34e − 04	+	**3.74e − 05 ± 6.00e − 05**	−
f_{15}	**2.68e + 02 ± 3.86e + 01**	6.57e + 02 ± 1.10e + 02	+	1.33e + 03 ± 2.32e + 02	+	4.14e + 02 ± 1.88e + 02	+
f_{16}	**2.35e + 00 ± 8.17e − 01**	1.71e + 01 ± 4.46e + 00	+	2.74e + 01 ± 4.27e + 00	+	1.25e + 01 ± 1.79e + 00	+
f_{17}	7.93e + 00 ± 3.86e + 00	3.42e + 00 ± 4.83e − 01	−	8.65e + 00 ± 1.62e + 00	+	**7.23e − 02 ± 4.95e − 02**	−
f_{18}	1.65e + 01 ± 9.86e + 00	1.21e + 01 ± 1.68e + 00	−	3.30e + 01 ± 6.61e + 00	+	**5.77e − 01 ± 1.89e − 01**	−
f_{19}	**1.66e + 00 ± 3.11e − 01**	2.42e + 00 ± 7.13e − 01	+	7.90e + 00 ± 1.29e + 00	+	2.16e + 00 ± 1.17e + 00	=
f_{20}	1.27e + 00 ± 1.47e − 01	2.11e + 00 ± 1.14e − 01	+	**4.95e − 01 ± 6.73e − 02**	−	1.54e + 00 ± 9.48e − 02	+
f_{21}	3.42e + 00 ± 3.50e + 00	4.15e + 00 ± 5.88e + 00	=	**3.36e + 00 ± 3.38e + 00**	=	4.76e + 00 ± 7.35e + 00	=
f_{22}	6.27e + 00 ± 7.42e + 00	7.94e + 00 ± 9.11e + 00	+	**5.11e + 00 ± 5.83e + 00**	−	7.58e + 00 ± 8.63e + 00	+
f_{23}	**6.62e − 01 ± 2.75e − 01**	2.47e + 00 ± 7.99e − 01	+	2.52e + 00 ± 4.22e − 01	+	7.85e − 01 ± 2.66e − 01	+
f_{24}	**3.11e + 02 ± 5.78e + 01**	3.72e + 02 ± 4.72e + 01	+	1.08e + 03 ± 1.53e + 02	+	3.13e + 02 ± 1.12e + 02	+

Table 12
SPAM against MDE-pBX, CCPSO2, MACh, on CEC2008 in 1000D.

	SPAM	MDE-pBX		CCPSO2		MACh	
f_1	**5.12e − 13 ± 6.32e − 13**	1.21e + 05 ± 4.41e + 04	+	1.01e − 11 ± 3.57e − 12	+	6.23e + 04 ± 6.12e + 05	+
f_2	**1.32e − 01 ± 3.27e − 02**	1.17e + 02 ± 4.09e + 00	+	4.10e + 01 ± 2.46e + 01	+	1.84e + 02 ± 1.81e + 00	+
f_3	**8.97e + 02 ± 2.83e + 01**	3.13e + 10 ± 1.65e + 10	+	1.41e + 03 ± 1.12e + 02	+	1.45e + 11 ± 1.01e + 12	+
f_4	**7.17e + 01 ± 1.16e + 01**	7.93e + 03 ± 2.55e + 02	+	1.30e + 02 ± 1.07e + 02	+	1.56e + 04 ± 1.02e + 03	+
f_5	**7.56e − 05 ± 7.36e − 04**	1.26e + 03 ± 4.60e + 02	+	8.59e − 04 ± 2.98e − 03	+	8.17e + 00 ± 1.09e + 01	+
f_6	1.22e + 01 ± 9.51e + 00	1.93e + 01 ± 5.10e − 02	+	**7.23e − 11 ± 9.34e − 11**	−	1.99e + 01 ± 3.19e − 01	+

Table 13
SPAM against MDE-pBX, CCPSO2, MACh, on CEC2010 in 1000D.

	SPAM	MDE-pBX		CCPSO2		MACh	
f_1	1.92e − 06 ± 1.33e − 07	1.05e + 09 ± 6.58e + 08	+	**6.47e − 14 ± 1.41e − 13**	−	2.45e + 11 ± 4.07e + 10	+
f_2	**6.15e + 01 ± 1.28e + 01**	7.02e + 03 ± 2.38e + 02	+	1.36e + 02 ± 1.11e + 02	+	1.97e + 04 ± 1.74e + 03	+
f_3	8.06e + 00 ± 9.49e + 00	1.93e + 01 ± 4.76e − 02	+	**7.34e − 11 ± 1.05e − 10**	−	2.02e + 01 ± 4.21e − 01	+
f_4	**1.14e + 11 ± 4.32e + 10**	3.21e + 12 ± 9.76e + 11	+	2.14e + 12 ± 1.27e + 12	+	1.24e + 15 ± 5.69e + 14	+
f_5	6.78e + 08 ± 9.14e + 07	**1.54e + 08 ± 2.77e + 07**	−	3.92e + 08 ± 7.98e + 07	−	8.09e + 08 ± 6.58e + 07	+
f_6	1.98e + 07 ± 7.24e + 04	**3.65e + 06 ± 1.75e + 06**	−	1.71e + 07 ± 4.45e + 06	−	2.03e + 07 ± 2.24e + 05	+
f_7	**4.17e + 05 ± 5.05e + 05**	6.79e + 06 ± 1.01e + 07	+	7.60e + 09 ± 9.72e + 09	+	7.64e + 11 ± 5.09e + 11	+
f_8	**3.17e + 06 ± 2.82e + 06**	2.03e + 08 ± 1.63e + 08	+	5.46e + 07 ± 4.16e + 07	+	5.34e + 16 ± 3.07e + 16	+
f_9	**3.21e + 06 ± 3.48e + 05**	1.68e + 09 ± 1.00e + 09	+	5.01e + 07 ± 7.68e + 06	+	2.81e + 11 ± 4.08e + 10	+
f_{10}	5.37e + 03 ± 3.84e + 02	7.33e + 03 ± 2.55e + 02	+	**4.57e + 03 ± 2.75e + 02**	−	1.85e + 04 ± 1.76e + 03	+
f_{11}	2.00e + 02 ± 2.22e + 01	2.06e + 02 ± 2.40e + 00	+	**2.00e + 02 ± 5.98e + 00**	−	2.20e + 02 ± 3.44e + 00	+
f_{12}	**3.76e − 03 ± 1.40e − 03**	2.92e + 05 ± 6.60e + 04	+	6.12e + 04 ± 8.14e + 04	+	1.69e + 07 ± 4.07e + 06	+
f_{13}	1.22e + 03 ± 8.23e + 02	2.88e + 09 ± 3.17e + 09	+	**1.14e + 03 ± 5.42e + 02**	=	1.27e + 12 ± 5.77e + 11	+
f_{14}	**4.65e + 06 ± 4.18e + 05**	1.04e + 09 ± 1.97e + 08	+	1.60e + 08 ± 3.35e + 07	+	3.23e + 11 ± 3.62e + 10	+
f_{15}	1.00e + 04 ± 1.04e + 03	**7.44e + 03 ± 2.80e + 02**	−	9.31e + 03 ± 5.52e + 02	−	1.86e + 04 ± 1.58e + 03	+
f_{16}	**2.64e + 02 ± 1.26e + 02**	3.84e + 02 ± 1.22e + 00	=	3.95e + 02 ± 1.45e + 00	=	4.00e + 02 ± 7.55e + 00	+
f_{17}	**4.36e − 01 ± 1.13e − 01**	4.35e + 05 ± 8.33e + 04	+	1.41e + 05 ± 1.44e + 05	+	4.64e + 07 ± 1.30e + 07	+
f_{18}	**1.71e + 03 ± 1.09e + 03**	3.73e + 10 ± 1.95e + 10	+	5.62e + 03 ± 4.13e + 03	+	5.38e + 12 ± 7.78e + 11	+
f_{19}	**3.32e + 05 ± 3.25e + 04**	9.22e + 05 ± 1.06e + 05	+	1.14e + 06 ± 1.22e + 06	+	1.24e + 08 ± 3.39e + 07	+
f_{20}	**8.92e + 02 ± 3.28e + 01**	4.18e + 10 ± 2.02e + 10	+	1.42e + 03 ± 1.19e + 02	+	6.07e + 12 ± 8.88e + 11	+

Table 14
Holm test on the Fitness, reference algorithm = SPAM (Rank = 3.17e+00).

j	Optimizer	*Rank*	$z_j \pm p_j \pm \delta/j$	Hypothesis		
1	MACh	2.31e+00	−6.95e+00	1.77e−12	5.00e−02	Rejected
2	MDE-pBX	2.29e+00	−7.14e+00	4.69e−13	2.50e−02	Rejected
3	CCPSO2	2.21e+00	−7.75e+00	4.43e−15	1.67e−02	Rejected

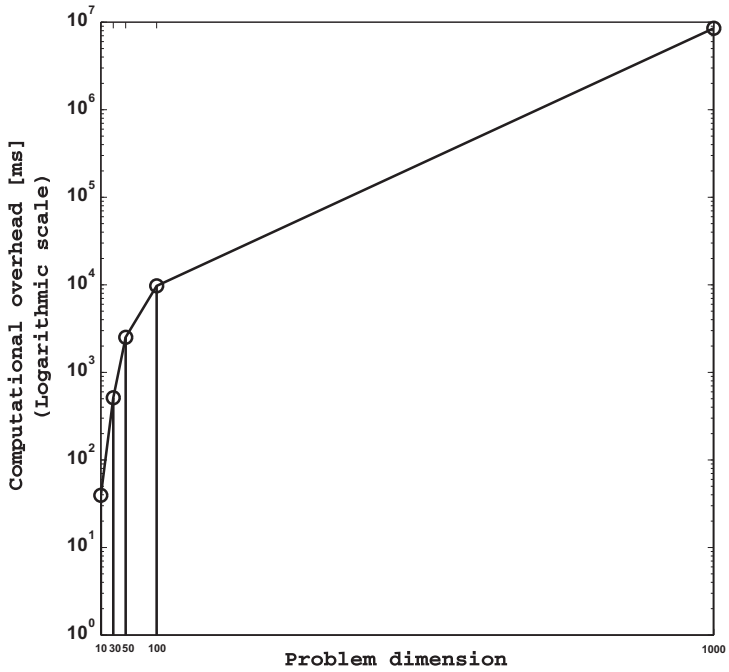

Fig. 9. Computational cost of analysis on separability.

- Compact Particle Swarm Optimization (CPSO) proposed in [55] with virtual population size of 300 individuals, $\phi_1 = -0.2, \phi_2 = -0.07, \phi_3 = 3.74, \gamma_1 = 1$, and $\gamma_2 = 1$.
- Covariance Matrix Adaptation Evolution Strategy with restart (G-CMAES) proposed in [84] with factor for increasing the population size equal to 2.
- Self-Adaptive Differential Evolution proposed in [8] with learning period equal to 50 and population size equal to 50 individuals.
- Adaptive Differential Evolution (JADE) proposed in [85] with population size equal to 60 individuals, group size factor $p = 0.05$ and parameters adaptation rate factor $c = 0.1$.
- Parallel Memetic Structure (PMS) proposed in [56]. For this scheme α_e has been set equal to 0.95, and the local search S is performed on all the variables for 150 iterations with a starting exploratory radius ρ set to the 40% of the search space width. Regarding the local search R, h is a vector of $0.1, \alpha = 2, \beta = 0.5$, while ε has been set equal to 10^{-5}.
- Cultural Algorithms – Iterated Local Search (CA-ILS) proposed in [38] with population size $\mu = 3$, maximum moves per individual $\eta = 5$, acceptance rate $\rho = 0.3$, initial move length proportion $\tau = 0.02$, and move length constraint $s = 0$. The reduction rate β for the global temperature T have been set equal to 0.838 and 100 respectively. As local search, in this study we made use of a stochastic technique namely Simultaneous Perturbation Stochastic Search (SPSA) [86] with $a = 0.5, A = 1, \alpha = 0.602, c = 0.032, \gamma = 0.1$ and $\epsilon = 0.01$.

All the detailed results are not reported in this paper in order to avoid an excessive paper length.[2] However, as summary of the experiments, this subsection shows the result of the Holm–Bonferroni procedure applied to the entire data set. By following the Holm–Bonferroni procedure, the 12 algorithms under analysis have been ranked on the basis of their average performance calculated over the 132 test problems. More specifically, a score R_i for $i = 1, \ldots, N_A$ (where N_A is the number of algorithms under analysis, $N_A = 12$ in our case) has been assigned. The score has been assigned in the following way: for each problem, a score of 12 is assigned to the algorithm displaying the best performance, 11 is assigned to the second best, 9 to the third and so on. The

[2] The detailed results are available at the url https://sites.google.com/site/facaraff/home/Downloads/SPAM_Detailed_Results.pdf?attredirects=0&d=1.

algorithm displaying the worst performance scores 1. For each algorithm, the scores obtained on each problem are summed up averaged over the amount of test problems (132 in our case). On the basis of these scores the algorithms are sorted (ranked). With the calculated R_i values, PMS has been taken as a reference algorithm. Indicating with R_0 the rank of PMS, and with R_j for $j = 1, \ldots, N_A - 1$ the rank of one of the remaining eleven algorithms, the values z_j have been calculated as

$$z_j = \frac{R_j - R_0}{\sqrt{\frac{N_A(N_A+1)}{6N_{TP}}}} \tag{13}$$

where N_{TP} is the number of test problems in consideration ($N_{TP} = 132$ in our case). By means of the z_j values, the corresponding cumulative normal distribution values p_j have been calculated. These p_j values have then been compared with the corresponding δ/j where δ is the level of confidence, set to 0.05 in our case. Table 15 displays the ranks, z_j values, p_j values, and corresponding δ/j obtained in this way. The rank of SPAM is shown in parenthesis. Moreover, it is indicated whether the null-hypothesis (that the two algorithms have indistinguishable performances) is "Rejected", i.e. SPAM statistically outperforms the algorithm under consideration, or "Accepted" if the distribution of values can be considered the same (there is no outperformance).

Table 15
Holm test on the fitness, reference algorithm = SPAM (Rank = 9.60e+00).

j	Optimizer	*Rank*	$z_j \pm p_j \pm \delta/j$	Hypothesis		
1	SADE	7.43e+00	−5.31e+00	5.57e−08	5.00e−02	Rejected
2	MDE-pBX	7.38e+00	−5.44e+00	2.71e−08	2.50e−02	Rejected
3	CCPSO2	7.24e+00	−5.77e+00	3.94e−09	1.67e−02	Rejected
4	PMS	7.08e+00	−6.16e+00	3.62e−10	1.25e−02	Rejected
5	CLPSO	6.92e+00	−6.57e+00	2.53e−11	1.00e−02	Rejected
6	MACh	6.84e+00	−6.75e+00	7.16e−12	8.33e−03	Rejected
7	G-CMAES	6.52e+00	−7.53e+00	2.46e−14	7.14e−03	Rejected
8	SPAM05	6.46e+00	−7.68e+00	7.80e−15	6.25e−03	Rejected
9	JADE	6.20e+00	−8.33e+00	3.98e−17	5.56e−03	Rejected
10	cPSO	3.70e+00	−1.45e+01	1.15e−47	5.00e−03	Rejected
11	CA-ILS	2.40e+00	−1.76e+01	7.39e−70	4.55e−03	Rejected

Table 16
Average fitness ± standard deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against DE-pBX, CCPSO2 and MACh on LJP problem in 120 dimensions.

SPAM	MDE-pBX		CCPSO2		MACh	
$\mathbf{-9.29e+01 \pm 8.54e+00}$	$-8.25e+01 \pm 6.12e+00$	=	$-2.35e+01 \pm 1.43e+00$	+	$-1.16e+01 \pm 1.72e+00$	+

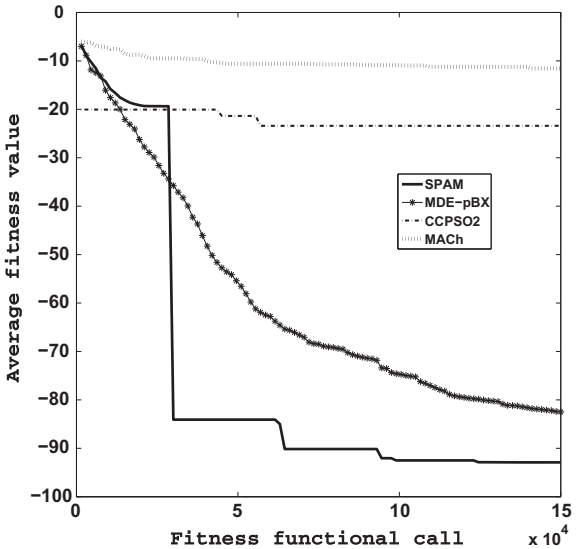

Fig. 10. Average fitness trend of SPAM, DE-pBX, CCPSO2 and MACh for the LJP problem.

It can easily be observed that also according the Holm–Bonferroni procedure the proposed SPAM by far outperforms all the other algorithms contained in this study.

4.5. A real world problem: Lennard–Jones Potential minimization

In order to prove the viability of the proposed approach over real-world problems, SPAM as well as MDE-pBX, CCPSO2, and MACh have been run over the Lennard–Jones Potential (LJP) problem. The LJP problem is widely used in physics for representing the interaction energy between non-bonding particles in a fluid. In this case, according to the description given in CEC2011 [87], we refer to the minimization problem of the potential energy of a set of atoms by locating them within the three-dimensional space. In order to determine their position and evaluate the relative LJP 3 parameters, i.e. coordinates along the axes x, y, and z, for each atom need to be stored in the fitness function. In this study, an atomic cluster of 40 atoms has been considered, resulting into a 120-dimensional problem. Each algorithm under examination has been run 10 times with a computational budget of 150,000 functional calls (in accordance with the description given in [87]). The parameters of the algorithms are the same used above. As for the CCPSO2, we adapted the problem divisions to the dimensionality of the specific case, i.e. we used 30, 60 divisions. Numerical results are given in Table 16 and Fig. 10, respectively.

Numerical results for this real-world application show that also in this case SPAM can reliably detect solutions characterized by a high performance.

5. Conclusion

This paper proposes a prototype for automatic design of optimization algorithms. This computational prototype is based on a problem analyser that estimates the degree of separability of the optimization problem by measuring the correlation between pairs of variables. Subsequently, the result of this analysis is manipulated to determine the activation probabilities of two operators connected in a Parallel Memetic Structure. The first performs moves along the axes and thus is suitable to address separable problems while the second simultaneously perturbs all the variables and follows the direction of the gradient thus being suitable for non-separable problems. The degree of separability of a problem is here interpreted, from an optimization point of view, as a fuzzy property that can vary within two extreme conditions. Numerical results show that, despite being at an early stage, the system is extremely promising as it appears to robustly tailor the algorithm to each problem and outperform modern algorithms based on adaptive schemes and multiple operators. This result is especially important if we consider that the algorithmic structure itself is extremely simple as inspired by the Ockham's Razor in Memetic Computing.

The main novelty of the proposed approach is that SPAM, albeit simple, is designed/tuned by a machine on the basis of a problem analysis. This fact allows a lower experience based human influence and a higher machine automatism/intelligence. This paper must be interpreted as a first step towards a complete software platform for automatic design of optimization algorithms. Future works on separability will consider other ways to measure the correlation between pairs of variables (usign e.g. Spearman coefficient). More importantly, future works will extend the analysis to other problem properties, such as ill-conditioning and multi-modality. In addition, future work will investigate broader databases of operators and other design criteria.

Acknowledgements

This research is supported by the Academy of Finland, Akatemiatutkija 130600, Algorithmic Design Issues in Memetic Computing. The numerical experiments have been carried out on the computer network of the De Montfort University by means of the software for distributed optimization Kimeme [88] within the MemeNet Project.

References

[1] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computation, Springer Verlag, Berlin, 2003. pp. 175–188.
[2] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[3] A. Auger, O. Teytaud, Continuous lunches are free!, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, ACM, 2007, pp 916–922.
[4] R. Poli, M. Graff, There is a free lunch for hyper-heuristics, genetic programming and computer scientists, in: EuroGP, 2009, pp. 195–207.
[5] T. Chabuk, J. Reggia, J. Lohn, D. Linden, Causally-guided evolutionary optimization and its application to antenna array design, Int. Comput. Aided Eng. 19 (2012) 111–124.
[6] N. Salvatore, A. Caponio, F. Neri, S. Stasi, G.L. Cascella, Optimization of delayed-state Kalman filter-based algorithm via differential evolution for sensorless control of induction motors, IEEE Trans. Ind. Electron. 57 (1) (2010) 385–394.
[7] H. Tao, J.M. Zain, M.M. Ahmed, A.N. Abdalla, W. Jing, A wavelet-based particle swarm optimization algorithm for digital image watermarking, Int. Comput.-Aided Eng. 19 (1) (2012) 81–91.
[8] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[9] R. Mallipeddi, S. Mallipeddi, P.N. Suganthan, Ensemble strategies with adaptive evolutionary programming, Inform. Sci. 180 (9) (2010) 1571–1581.
[10] J. Brest, M.S. Maucec, Self-adaptive differential evolution algorithm using population size reduction and three strategies, Soft Comput. 15 (11) (2011) 2157–2174.
[11] J. Brest, P. Korosec, J. Silc, A. Zamuda, B. Boskovic, M.S. Maucec, Differential evolution and differential ant-stigmergy on dynamic optimisation problems, Int. J. Syst. Sci. 44 (4) (2013) 663–679.

[12] Z. Zhu, J. Zhou, Z. Ji, Y.-H. Shi, DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm, IEEE Trans. Evol. Comput. 15 (5) (2011) 643–658.
[13] J.A. Vrugt, B.A. Robinson, J.M. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, IEEE Trans. Evol. Comput. 13 (2) (2009) 243–259.
[14] F. Peng, K. Tang, G. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization, IEEE Trans. Evol. Comput. 14 (5) (2010) 782–800.
[15] L. Xu, F. Hutter, H. Hoos, K. Leyton-Brown, SATzilla: portfolio-based algorithm selection for SAT, J. Artif. Intell. Res. 32 (2008) 565–606.
[16] H.H. Hoos, Programming by optimization, Commun. ACM 55 (2) (2012) 70–80.
[17] F. Hutter, H.H. Hoos, K. Leyton-Brown, Tradeoffs in the empirical evaluation of competing algorithm designs, Ann. Math. Artif. Intell. 60 (1–2) (2010) 65–89.
[18] F. Hutter, L. Xu, H.H. Hoos, K. Leyton-Brown, Algorithm runtime prediction: methods & evaluation, Artif. Intell. 206 (2014) 79–111.
[19] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling: a sales summit, in: Proceedings of the Third International Conference on Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 2079, Springer, 2000, pp. 176–190.
[20] E.K. Burke, G. Kendall, E. Soubeiga, A tabu search hyperheuristic for timetabling and rostering, J. Heuris. 9 (6) (2003) 451–470.
[21] E. Özcan, B. Bilgin, E.E. Korkmaz, A comprehensive analysis of hyper-heuristics, Intell. Data Anal. 12 (1) (2008) 3–23.
[22] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, J. Woodward, Classification of hyper-heuristic approaches, in: Handbook of Meta-Heuristics, Springer, 2010, pp. 449–468.
[23] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyperheuristic for educational timetabling problems, Euro. J. Operat. Res. 176 (2007) 177–192.
[24] G. Kendall, P. Cowling, E. Soubeiga, Choice function and random hyperheuristics, in: Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning, 2002, pp. 667–671.
[25] K.A. Dowsland, E. Soubeiga, E. Burke, A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation, Euro. J. Operat. Res. 179 (3) (2007) 759–774.
[26] E.K. Burke, Y. Bykov, in: PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling. <http://w1.cirrelt.ca/patat2008/PATAT7PROCEEDINGS/Pape>.
[27] G. Acampora, M. Gaeta, V. Loia, Hierarchical optimization of personalized experiences for e-Learning systems through evolutionary models, Neural Comput. Appl. 20 (5) (2011) 641–657.
[28] G. Acampora, J.M. Cadenas, V. Loia, E.M. Ballester, A multi-agent memetic system for human-based knowledge selection, IEEE Trans. Syst. Man Cybernet. Part A 41 (5) (2011) 946–960.
[29] F. Neri, C. Cotta, P. Moscato, Handbook of Memetic Algorithms, Studies in Computational Intelligence, vol. 379, Springer, 2011.
[30] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: a literature review, Swarm Evol. Comput. 2 (2012) 1–14.
[31] H.-B. Duan, C.-F. Xu, Z.-H. Xing, A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems, Int. J. Neural Syst. 20 (1) (2010) 39–50.
[32] P. Moscato, M. Norman, A Competitive and Cooperative Approach to Complex Combinatorial Search, Tech. Rep. 790, 1989.
[33] P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Tech. Rep. 826, 1989.
[34] D. Molina, M. Lozano, C. Garcia-Martinez, F. Herrera, Memetic algorithms for continuous optimization based on local search chains, Evol. Comput. 18 (1) (2010) 27–63.
[35] G. Iacca, F. Neri, E. Mininno, Y.S. Ong, M.H. Lim, Ockham's razor in memetic computing: three stage optimal memetic exploration, Inform. Sci. 188 (2012) 17–43.
[36] A. Caponio, G.L. Cascella, F. Neri, N. Salvatore, M. Sumner, A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives, IEEE Trans. Syst. Man Cybernet. – Part B 37 (1) (2007) 28–41.
[37] N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, IEEE Trans. Evol. Comput. 9 (5) (2005) 474–488.
[38] T.T. Nguyen, X. Yao, An experimental study of hybridizing cultural algorithms and local search, Int. J. Neu. Syst. 18 (1) (2008) 1–17.
[39] J.E. Smith, Coevolving memetic algorithms: a review and progress report, IEEE Trans. Syst. Man Cybernet. Part B 37 (1) (2007) 6–17.
[40] J.E. Smith, Estimating meme fitness in adaptive memetic algorithms for combinatorial problems, Evol. Comput. 20 (2) (2012) 165–188.
[41] Y.-S. Ong, M.-H. Lim, X. Chen, Research Frontier: Towards Memetic Computing, Tech. Rep. C2i–1209, School of Computer Engineering, Nanyang Technological University, Singapore, 2009.
[42] Y.-S. Ong, M.-H. Lim, X. Chen, Memetic computation-past, present and future, IEEE Comput. Intell. Mag. 5 (2) (2010) 24–31.
[43] R. Meuth, M.H. Lim, Y.S. Ong, D.C. Wunsch-II, A proposition on memes and meta-memes in computing for higher-order learning, Memetic Comput. J. 1 (2) (2009) 85–100.
[44] Z. Zhu, S. Jia, Z. Ji, Towards a memetic feature selection paradigm, IEEE Comput. Intell. Mag. 5 (2) (2010) 41–53.
[45] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, G. Rudolph, Exploratory landscape analysis, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, 2011, pp. 829–836.
[46] Y. Wu, J.A.W. McCall, D. Corne, O. Regnier-Coudert, Landscape analysis for hyperheuristic Bayesian Network structure learning on unseen problems, in: IEEE Congress on Evolutionary Computation, 2012.
[47] Z. Ren, H. Jiang, J. Xuan, Z. Luo, Hyper-heuristics with low level parameter adaptation, Evol. Comput. 20 (2) (2012) 189–227.
[48] Y. Hamadi, E. Monfroy, F. Saubion, Autonomous Search, Springer, 2012.
[49] S.L. Epstein, S. Petrovic, Learning a mixture of search heuristics, in: Y. Hamadi, E. Monfroy, F. Saubion (Eds.), Autonomous Search, Springer, 2012, pp. 97–127.
[50] R. Battiti, P. Campigotto, An investigation of reinforcement learning for reactive search optimization, in: Y. Hamadi, E. Monfroy, F. Saubion (Eds.), Autonomous Search, Springer, 2012, pp. 131–160.
[51] J. Maturana, A. Fialho, F. Saubion, M. Schoenauer, F. Lardeux, M. Sebag, Adaptive operator selection and management in evolutionary algorithms, in: Y. Hamadi, E. Monfroy, F. Saubion (Eds.), Autonomous Search, Springer, 2012, pp. 161–189.
[52] Y. Hamadi, S. Jabbour, J. Sais, Control-based clause sharing in parallel SAT solving, in: Y. Hamadi, E. Monfroy, F. Saubion (Eds.), Autonomous Search, Springer, 2012, pp. 245–267.
[53] M. Birattari, Z. Yuan, P. Balaprakash, T. Stützle, F-Race and iterated F-Race: an overview, in: T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss (Eds.), Experimental Methods for the Analysis of Optimization Algorithms, Springer, 2010.
[54] M. Lopez-Ibanez, J. Dubois-Lacoste, T. Stützle, M. Birattari, The Irace Package, Iterated Race for Automatic Algorithm Configuration, Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Universite Libre de Bruxelles, Belgium, 2011.
[55] F. Neri, E. Mininno, G. Iacca, Compact particle swarm optimization, Inform. Sci. 239 (2013) 96–121.
[56] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Inform. Sci. 227 (0) (2013) 60–82. ISSN 0020-0255.
[57] N. Hansen, A. Auger, S. Finck, R. Ros, et al., Real-Parameter Black-Box Optimization Benchmarking 2010: Noiseless Functions Definitions, Tech. Rep. RR-6829, INRIA, 2010.
[58] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evol. Comput. 11 (1) (2003) 1–18.
[59] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 1996, pp. 312–317.
[60] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159–195.
[61] S.-F. Lin, Y.-C. Cheng, A separability detection approach to cooperative particle swarm optimization, 2011, pp. 1141–1145.

[62] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Tech. Rep. 2005005, Nanyang Technological University and KanGAL, Singapore and IIT Kanpur, India, 2005.
[63] G. Harik, Linkage Learning via Probabilistic Modeling in the ECGA, Tech. Rep. 99010, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.
[64] H. Li, Q. Zhang, A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages, in: Parallel Problem Solving from Nature – PPSN IX, Lecture Notes in Computer Science, vol. 4193, Springer, 2006, pp. 583–592.
[65] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, Linkage problem, distribution estimation, and bayesian networks, Evol. Comput. 8 (2000) 311–340. ISSN 1063-6560.
[66] G.R. Harik, F.G. Lobo, K. Sastry, Linkage Learning via Probabilistic Modeling in the Extended Compact Genetic Algorithm (ECGA), in: M. Pelikan, K. Sastry, E. Cantú-Paz (Eds.), Scalable Optimization via Probabilistic Modeling, Studies in Computational intelligence, vol. 33, Springer, 2006, pp. 39–61.
[67] P.L. Lanzi, L. Nichetti, K. Sastry, D.E. Goldberg, Real-coded extended compact genetic algorithm based on mixtures of models, in: Linkage in Evolutionary Computation, Studies in Computational Intelligence, vol. 157, Springer, 2008, pp. 335–358.
[68] L.-Y. Tseng, C. Chen, Multiple trajectory search for large scale global optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2008, pp. 3052–3059.
[69] H.H. Rosenbrock, An automatic method for finding the greatest or least value of a function, Comput. J. 3 (3) (1960) 175–184.
[70] M.S. Bazaraa, H.D. Sherali, C.M. Shetty, Nonlinear Programming: Theory And Algorithms, Wiley-Interscience, 2006. ISBN 0-471-78777-9.
[71] D. Zaharie, Influence of crossover on the behavior of differential evolution algorithms, Appl. Soft Comput. 9 (3) (2009) 1126–1138. ISSN 1568-4946.
[72] F. Neri, G. Iacca, E. Mininno, Disturbed exploitation compact differential evolution for limited memory optimization problems, Inform. Sci. 181 (12) (2011) 2469–2487.
[73] J.J. Liang, B.Y. Qu, P.N. Suganthan, A.G. Hernndez-Daz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization, Tech. Rep. 201212, Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore, 2013.
[74] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang, Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization, Tech. Rep., Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.
[75] 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, Tech. Rep., University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL): Hefei, Anhui, China, 2010.
[76] F. Wilcoxon, Individual comparisons by ranking methods, Biomet. Bull. 1 (6) (1945) 80–83.
[77] N. Hansen, The CMA Evolution Strategy, 2012. <http://www.lri.fr/hansen/cmaesintro.html>.
[78] S. Islam, S. Das, S. Ghosh, S. Roy, P. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 42 (2) (2012) 482–500.
[79] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2012) 210–224.
[80] D. Molina, M. Lozano, F. Herrera, MA-SW-Chains: memetic algorithm based on local search chains for large scale continuous global optimization, in: IEEE Congress on Evolutionary Computation, 2010b, pp. 1–8.
[81] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (2) (1979) 65–70.
[82] S. Garcia, A. Fernandez, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2008) 959–977.
[83] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[84] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2005, pp. 1769–1776.
[85] J. Zhang, A. Sanderson, JADE: adaptive differential evolution with optional external archive, EEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[86] J. Spall, A stochastic approximation technique for generating maximum likelihood parameter estimates, in: American Control Conference, 1987, 1987, pp. 1161–1167.
[87] S. Das, P. Suganthan, Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India.
[88] Cyber Dyne Srl Home Page, Kimeme, 2012. <http://cyberdynesoft.it/>.