A Separability Prototype for Automatic Memes with Adaptive Operator Selection

Michael G. Epitropakis*, Fabio Caraffini†, Ferrante Neri† and Edmund K. Burke*
*Computing Science and Mathematics, School of Natural Sciences, University of Stirling,
Stirling FK9 4LA, Scotland, United Kingdom
†Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University,
The Gateway, Leicester LE1 9BH, England, United Kingdom,
Email: {mge@cs.stir.ac.uk, fabio.caraffini@dmu.ac.uk, fneri@dmu.ac.uk, e.k.burke@stir.ac.uk}

***Abstract*—One of the main challenges in algorithmics in general, and in Memetic Computing, in particular, is the automatic design of search algorithms. A recent advance in this direction (in terms of continuous problems) is the development of a software prototype that builds up an algorithm based upon a problem analysis of its separability. This prototype has been called the Separability Prototype for Automatic Memes (SPAM). This article modifies the SPAM by incorporating within it an adaptive model used in hyper-heuristics for tackling optimization problems. This model, namely Adaptive Operator Selection (AOS), rewards at run time the most promising heuristics/memes so that they are more likely to be used in the following stages of the search process. The resulting framework, here referred to as SPAM-AOS, has been tested on various benchmark problems and compared with modern algorithms representing the-state-of-the-art of search for continuous problems. Numerical results show that the proposed SPAM-AOS is a promising framework that outperforms the original SPAM and other modern algorithms. Most importantly, this study shows how certain areas of Memetic Computing and Hyper-heuristics are very closely related topics and it also shows that their combination can lead to the development of powerful algorithmic frameworks.**

I. Introduction

The No Free Lunch Theorems (NFLTs) [1] state that, under certain hypotheses, the performance of any pair of algorithms A and B averaged over all possible problems is the same. The hypotheses of the NFLTs are that the algorithms are non-revisiting and that the decision space is discrete. The satisfaction of these hypotheses is often non-realistic. For example, Auger and Teytaud [2] show that NFLTs are not valid for continuous problems while Poli and Graff [3] show that the NFLTs are not valid in meta-spaces, i.e. the space of operators. Nonetheless, NFLTs has been an extremely important result for the computer science community as it changed the landscape of search methodology research. More specifically, until the 1990s, many researchers were attempting to define a "super-algorithm" that is an algorithm which is superior to all the other algorithms over all problems, see e.g. [4]. After the publication of the NFLTs, researchers had to radically change their way of thinking. The search algorithm became a domain specific solver. The problem can be seen as the starting point for an algorithmic design process whose outcome is a procedure that addresses the features of that given problem, see e.g. [5], [6], and [7].

Since a problem change would result in a new design (and thus human effort), some research has been oriented towards a design of "flexible" search algorithm, i.e. those algorithms that change their features and adapt to (usually slightly) diverse problems hence displaying a reasonable good performance on an array of problems without the need of a major human intervention from the algorithmic designer, see [8]. This flexibility is usually achieved by means of the use of multiple algorithmic operators and an adaptive system that coordinates these operators, see e.g. [9] and [10]. The main idea is that multiple diverse algorithmic operators compensate for each other with their search logics. In this way, the algorithmic framework that employs multiple operators can reliably and robustly tackle various problem features, thus adapting to a new problem. This idea is the backbone of two algorithmic philosophies, Hyper-heuristics and Memetic Computing.

A hyper-heuristic is an algorithm composed of multiple algorithms coordinated by another software component. Usually, the latter incorporates a machine learning technique and acts as a supervisor structure that learns which algorithms are the most suitable for a given problem. An extensive literature on this topic is available, for example, see [11] and [12]. More recently, graph colouring heuristics have been hybridized with a random ordering heuristic [13]. Extensive discussions about advances in hyper-heuristics are given in [14]–[17]. The most challenging part of the hyper-heuristic design is the logic behind the coordination of the algorithms. A classical approach consists of assigning a score and rewarding the most promising heuristics: the so called choice function [11]. Recently, a wide range of diverse approaches have been proposed. Many of these approaches are based on reinforcement learning in a stand alone and combined fashion, see for example [12], [18]–[20] and [21]. Reinforcement learning, making use of memory-based mechanisms has also been proposed [22]. Modern hyper-heuristics also make use of multi-agent operators, see [23] and [24]. A competition for hyper-heuristic development was held in 2011 based on a framework called HyFlex [25]. This competition generated several effective hyper-heuristic methods[1], see for example [26].

The concept of the hyper-heuristic is explored also under the name of algorithm portfolios where the emphasis is placed upon the diversity of the available algorithms that are available to compose the entire framework. Algorithms of this kind can be designed using heuristic rules for the coordination, see [27]

[1]See http://www.asap.cs.nott.ac.uk/external/chesc2011/

and [28]. A famous portfolio platform oriented towards the solution of the propositional satisfiability problem is called SATzilla, see [29] and [30]. Among the several studies that have been carried out on this platform, we highlight here the work on the automatic coordination system in [31] and on the prediction of the algorithm run time [32].

Memetic Computing (MC) is an area that investigates algorithmic structures that include multiple interacting heterogeneous operators. The interaction (coordination) amongst operators allows the proper functioning of the entire framework, see [33] and [8]. Clearly, there is a major overlap between hyper-heuristics and many areas of MC. However, the two concepts can be distinguished for a philosophical point that has some important implications in the algorithmic implementation. While a hyper-heuristic can be seen as a list of search methods and a component performing their coordination, a MC algorithm can be viewed as a whole algorithm composed of many parts in an unspecified way. In addition, in a hyper-heuristic, a unit representing the external framework can be thought of as a 'whole' search method with a 'budget'. In contrast, the unit of a MC structure can be seen as an algorithmic operator, that is a tuple composed of search move and a selection logic, see [34] and [35]. In other words, the unit representing many MC structures can (as a generalized statement) be viewed as being much 'smaller' than the unit composing a hyper-heuristic.

In many areas of MC, as well as for hyper-heuristics, the coordination between operators is the hardest and most challenging aspect in guaranteeing a good algorithmic performance. An example of meme coordination within memetic frameworks is meta-Lamarckian learning, [36], where the local search components are rewarded on the basis of their success history. In [37] [38], and [39], the coordination of the operators relies on evolution. The operators are encoded within the solution (self-adaptation) or in in a parallel population linked to the solutions (co-evolution). The operators related to the fittest individuals are selected for the subsequent generations. Another way to perform the coordination of the operators is by means of a control on the population diversity or its estimates [40]–[42].

All these adaptive optimization algorithms coordinate the sub-algorithms at runtime on the basis of feedback given by the algorithmic functioning. In most cases, this feedback is based on the success of the sub-algorithm/local search/algorithmic operator. Recent studies in MC propose a different logic: the algorithm is automatically built up on the basis of an analysis carried out on the problem and aiming at extracting its features. These studies at first focused on elementary structures, namely sequential and parallel [43], [44]. Subsequently a first working prototype of automatic design based on problem analysis restricted to the separability, namely Separability Prototype for Automatic Memes (SPAM), was proposed in [35].

This paper, whilst considering the algorithmic operators in [35], proposes to coordinate them by means of a hyper-heuristic rule. More specifically, we adopt the general concept of adaptive operator selection [18], [45]–[48] and incorporate it in the core structure of the SPAM algorithm. In this way, the resulting framework has the structure and operators of SPAM but, at run time, the selection of the components is intended to adapt progressively , on the basis of a success criterion, to the requirements of the problem.

The remainder of this paper is structured in the following way. Section II briefly illustrates the logic and the structure of the SPAM. Section III describes the proposed modified SPAM with hyper-heuristic coordination of the operators. Section IV displays the comparative results and the performance of the proposed algorithm with respect to the state-of-the-art. Section V gives the conclusions of this work.

II. Separability Prototype for Automatic Memes

The SPAM algorithm is a prototype of a large and ambitious project, that is a software platform for the automatic design of search methods in continuous optimization. This platform, currently under development, will analyze a given problem, in order to extract its features. These features will then be used to design the algorithm by selecting, combining, and linking the suitable algorithmic operators.

Before entering into the implementation details, let us define the notation used in this paper. Without loss of generality, we will 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.

The SPAM algorithm is composed of two local search operators connected in parallel, see [44]. The first operator, here indicated with S, is a local search algorithm which deals with a single solution along its n axes, i.e. it separately perturbs each design variable. This meme can be viewed as a fairly straightforward hill-descent algorithm. It was previously used in [49] 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), $x_i^t = x_i - \xi$, 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: $x_i^t = x_i + \frac{\xi}{2}$. Again, $\mathbf{x^t}$ replaces $\mathbf{x}$ if it outperforms it. If there is no update, then the exploration was unsuccessful. In this case, the radius ξ is halved. This is repeated for all the design variables. It has to be noted that the initial value of radius ξ is usually fixed and proportionate to the range of optimization space bounds (here ξ in fixed to 0.4). For the sake of clarity, Algorithm 1 describes the working principles of the S operator.

The second operator is the Rosenbrock algorithm (R) [50]. This operator, has been shown to always converge to a local optimum, under specific conditions [51]. At the beginning of the process, R is similar to S as it explores each of the n directions, by perturbing the input solution $\mathbf{x}$ with an initial step size vector $\mathbf{h}$. A matrix $\mathbf{A}$ is initialized as the identity matrix. While ever new improvements are found, for $j = 1, 2, \ldots, n$, a new trial point $\mathbf{x^t}$ is generated by perturbing the i^{th} design variable of solution $\mathbf{x}$ in the following way:

$$x_i^t = x_i + h_j \cdot A_{i,j} \tag{1}$$

Algorithm 1 Pseudo-code of the S operator.

```
1: INPUT $\mathbf{x}$
2: while condition on the local computational budget do
3:     for $i = 1 : n$ do
4:         $x_i^t = x_i - \xi$
5:         if $f\left(\mathbf{x^t}\right) \leq f(\mathbf{x})$ then
6:             $\mathbf{x} = \mathbf{x^t}$
7:         else
8:             $x_i^t = x_i + \frac{\xi}{2}$
9:             if $f\left(\mathbf{x^t}\right) \leq f(\mathbf{x})$ then
10:                $\mathbf{x} = \mathbf{x^t}$
11:            end if
12:        end if
13:    end for
14:    if $\mathbf{x} == \mathbf{x^t}$ then
15:        $\xi = \frac{\xi}{2}$
16:    end if
17: end while
18: OUTPUT $\mathbf{x}$
```

for $i = 1, 2, \ldots, n$. When successful, $\mathbf{x}$ is updated and the step size is increased by a factor α ($h_j = \alpha \cdot h_j$), otherwise it is decreased by means of a factor β then the opposite direction will be considered ($h_j = -\beta \cdot h_j$). We repeat this procedure until an improvement of solution $\mathbf{x}$ is found. After every success has been determined and examined in each base direction, then the Gram-Schmidt orthogonalization procedure is used to rotate the coordinate system towards the approximated gradient. This operation yields an updated version of the matrix $\mathbf{A}$. After this, the step size vector $\mathbf{h}$ is reinitialized and the process is repeated, employing the rotated coordinate system, and changing the value of $\mathbf{x}$ according to Eq. (1). It is important to note that, the use of a rotated coordinate system means that the trial generation mechanism corresponds to a diagonal move by following the direction determined by the gradient. The termination criterion of R is determined by two conditions. The first criterion is based on the minimum element of the vector $\mathbf{h}$. The second criterion concerns the minimum difference between $\mathbf{x^t}$ and $\mathbf{x}$ design variables. In formal terms, R is run until the following condition is true: $\min(|\mathbf{h}|) > \varepsilon\ OR\ \min(|\mathbf{x^t} - \mathbf{x}|) > \varepsilon$, where $\min()$ is the minimum vector element. If there is no improvement at all, only the first condition is considered[2].

These two local search operators are clearly very diverse in terms of the search logic. Whilst S attempts to optimize the problem by perturbing the axes separately (moves along the axes), R attempts to follow the directions of the gradient, hence performing diagonal moves.

The SPAM is based on the idea that (due to their features), S is efficient for tackling separable problems whilst R should be used only when the problem is not separable. Furthermore, the separability of a problem is considered in [35] as a fuzzy property of optimization problems. More specifically, it is possible to consider a problem as being separable to a certain degree. In this light, a problem analyzer has been designed to extract the *degree of separability* of the given problem. The degree of separability is an index between 0 and 1 that results from the problem analysis phase. On the basis of this parameter, an activation probability is given to S and R. Thus, a fully separable problem (e.g. the sphere function) will be solved only by S activations. In contrast, in the case of a highly non separable problem, the entire budget or a large portion of it is devoted to R. In the case of intermediate features, the budget will be shared between the two local searchers.

The parameter representing the degree of separability is calculated, for each problem, at the beginning of the search, by letting the Covariance Matrix Adaptation Evolution Strategy (CMAES) [52] perform some optimization steps and then manipulate the estimated covariance matrix. More specifically, from the covariance matrix, the Pearson correlation matrix $|\rho|$ is computed, and its elements are averaged and normalized in order to extract the index that describes the separability of the problem. For implementation details, see [35].

III. The Proposed Framework

As discussed previously, a problem analysis might reveal useful information on selecting an appropriate operator for the problem at hand. However, the selection of the "most suitable" operator might be a very challenging task, especially in cases where the problem analysis is not insightful enough. To alleviate the problem of selecting the most "suitable" operator *a priori*, we incorporate an Adaptive Operator Selection (AOS) model in SPAM's structure to adaptively choose the most preferable operator at hand. This section briefly describes the main algorithmic concepts behind the proposed framework SPAM-AOS, (SPAM with an Adaptive Operator Selection model).

The SPAM-AOS framework incorporates two main concepts, a credit assignment module $\mathcal{C}$ and an adaptive operator selection model $\mathcal{M}$, or a selective hyper-heuristic. The main role of the former is to estimate the quality of the applied operators based on feedback from the search process, whilst the latter adopts the estimated qualities to adaptively select which search operator to apply at the current stage. Intuitively, the model will select the most successful operator based on the historical information that has been acquired by the search process until the current stage. Several operator selection models have been proposed in the literature which are inspired by similar concepts from different scientific fields. Each model exhibits different characteristics and dynamics that are based on the acquired feedback during the search process. The proposed framework is a general strategy that is able to adopt any of the aforementioned adaptive operator selection models. Representative examples of such models include: probability matching [47], [48], adaptive pursuit [47], [48], statistical based models like the multinomial distribution with history forgetting [45], [46], and reinforcement learning approaches [18], [53]. However, for simplicity (and due to space limitations), in this paper we incorporate the well-known and widely used Probability Matching model [47], [48]. Future work will include extensive comparisons between the most representative models.

The credit assignment module and the adaptive operator selection model used in this paper are elucidated in the following sections.

A. Credit assignment module

The main role of a credit assignment module is to assign a representative score, or credit, that rewards the most "suitable"

[2]Note that, the parameter values are fixed to $\alpha = 2, \beta = 0.5, \varepsilon = 10^{-5}$ and the initial values of h_j to $h_j = 0.1, j = 1, 2, \ldots, n$.

operator at the current search stage. For each available operator we adopt as credit the simple fitness improvement [18], [45] between the current solution x^p and the best solution ever found by the algorithm x^e (the elite solution). Thereby, the fitness improvement η_a of an operator a can be easily calculated according to:

$$\eta_a = |f(x^p) - f(x^e)| \tag{2}$$

Notice that the credit equals zero, if no improvement is achieved.

In general, various different reward approaches can be used at this point, such as the latest reward (instantaneous), the average or a ranked-based reward [18]. However some of them tend to be unstable and noisy estimations of credit due to the stochastic nature of the search process. To alleviate this drawback, we estimate the empirical quality (or credit) of an operator by utilizing the average value of a sliding window of its latest w rewards. In detail, suppose that we have a pool of κ available operators, $A = \{a_1, a_2, \ldots, a_\kappa\}$; let S_i be a set of the latest w fitness improvement rewards (η_{a_i}) achieved by the operator a_i during the time step t of the algorithm. We adopt as the final credit assignment value $r_{a_i}(t)$, the average reward of the sliding window S_{a_i}, which can be calculated according to:

$$r_{a_i}(t) = \frac{\sum_{j=1}^{|S_{a_i}|} S_{a_i}(j)}{|S_{a_i}|} \tag{3}$$

where $|S_{a_i}|$ denotes the cardinality of the set S_{a_i}.

B. Adaptive Operator Selection model: Probability Matching

We utilize, as the adaptive operator, selection model the simple and well-known Probability Matching (PM) [18], [47], [48]. Intuitively, PM updates the selection probability of a specific operator proportionally to its empirical quality with respect to the others. In detail, suppose we have a set of κ available operators $A = \{a_1, a_2, \ldots, a_\kappa\}$ and a selection probability vector $P(t) = \{p_1(t), p_2(t), \ldots, p_\kappa(t)\}$. Initially, all operators have equal probability of being selected ($p_i = 1/\kappa, \forall i \in \{1, 2, \ldots, \kappa\}$). After the application of an operator on a solution vector, a reward ($r_{a_i}(t)$) is calculated based on the credit assignment module. The environment in which the adaptation procedure operates is non-stationary and the estimation of the empirical quality can be more reliable if the newest rewards influence the empirical quality more than the older ones. Thereby, PM estimates the empirical quality $q_{a_i}(t)$ of the operator a_i according to the following relaxation mechanism:

$$q_{a_i}(t+1) = q_{a_i}(t) + \gamma(r_{a_i}(t) - q_{a_i}(t)) \tag{4}$$

where $\gamma \in (0, 1]$ is the adaptation rate (γ is fixed to 0.1) [18], [47], [48].

Based on the quality estimate value of the operator a_i, PM updates its selection probability (p_{a_i}) according to the following formula:

$$p_{a_i}(t+1) = p_{\min} + (1 - \kappa \cdot p_{\min}) \frac{q_{a_i}(t+1)}{\sum_{i=1}^{\kappa} q_{a_i}(t+1)} \tag{5}$$

where $p_{\min} \in [0, 1]$ denotes the minimal probability value of each operator, to ensure that the chances of applying each operator will not vanish [47], [48]. Having calculated the probabilities of all operators in the pool, we employ a stochastic selection procedure (a simple roulette wheel) to select which operator to apply.

Algorithm 2 The general SPAM-AOS algorithmic scheme

```
1: Perform problem analysis as in [35] and calculate $x^e$.
2: INPUT: an initial solution $x^e$, a credit assignment module $\mathcal{C}$, and an AOS model $\mathcal{M}$.
3: Initialize the AOS model $\mathcal{M}$ and the credit assignment module $\mathcal{C}$.
4: $x^p = x^e$
5: while the remaining budget is available do
6:     Select $k_{str}$ based on the AOS model $\mathcal{M}$
7:     if $k_{str} = 1$ then
8:         Apply the S operator to $x^p$
9:     else
10:        Apply the R operator to $x^p$
11:    end if
12:    Update credit assignment module $\mathcal{C}$ based on the fitness improvement of $x^e, x^p$. (based on Eqs. (2), (3))
13:    Update the AOS model $\mathcal{M}$ (based on Eqs. (4), (5) )
14:    if the operator succeeded at improving upon $x^e$ performance then
15:        $x^e = x^p$
16:    end if
17:    if the operator failed at improving upon $x^e$ performance and the selected operator is the same that failed then
18:        Perturb $x^p$ according to an exponential crossover perturbation [35].
19:        if $x^p$ has a better performance against $x^e$ then
20:            $x^e = x^p$
21:        end if
22:    end if
23: end while
```

C. SPAM with an adaptive operator selection model

Having defined the SPAM algorithm and the two main concepts of the proposed framework, we proceed with the description of the proposed approach.

More specifically, the algorithmic framework of SPAM-AOS is strongly based on SPAM's structure. The proposed general algorithmic framework is briefly demonstrated in Algorithm 2. Initially the separability problem analysis is performed (line 1 of Alg. 2) and the best located solution or *elite* solution, x^e, is used as the initial point of the algorithm (line 2 of Alg. 2). As previously described, the separability problem analysis involves the application of the CMAES algorithm for a limited budget of function evaluations. Next, an initialization process of the credit assignment module $\mathcal{C}$ and the AOS model $\mathcal{M}$ is performed, if necessary (line 3 of Alg. 2). $\mathcal{C}$ initializes the data structures for the sliding windows and $\mathcal{M}$ assigns equal probabilities to all the used operators. For each step of the algorithm, we employ an AOS model $\mathcal{M}$ to select which operator k_{str} to apply on the x^p solution. In the general case, we might have a pool of κ available search operators, i.e., $k_{str} \in \{1, 2, \ldots, \kappa\}$. As in SPAM, we incorporate the S and R operators ($\kappa = 2$) and select one of them accordingly (lines 6–11 of Alg. 2). The current version of SPAM-AOS incorporates the PM model described previously, which stochastically selects k_{str}. After the application of the k_{str} operator, we score it based on the credit assignment module $\mathcal{C}$, which appropriately rewards the most suitable operator. Here, we utilize a reward based on the fitness improvement between the resulting solution x^p and the elite solution x^e, (line 12 of Alg. 2). Having calculated the reward of the applied operator, we update its quality estimation value and the corresponding selection probability based on Eqs. (4), (5) (line 13 of Alg. 2). The remaining steps of the algorithm are similar to SPAM (lines 14–22 of Alg. 2), i.e., we update the elite solution if we have improved it and we apply an exponential crossover perturbation strategy, if the applied operator failed to improve the elite solution x^e [35].

IV. Numerical Results

The proposed SPAM-AOS has been compared with the original SPAM algorithm [35], the CCPSO2 [54], and the MDE-pBX [55]. Their effectiveness is assessed on two benchmark suites that include scalable problems with different characteristics: the recently proposed CEC 2013 [56] (28 benchmark functions) and BBOB 2010 benchmark suites [57] (25 benchmark functions). In this paper, we consider the 10, 30 and 50-dimensional versions of the CEC 2013 suite and the 100-dimensional versions of the BBOB 2010 suite. A detailed description of the benchmarks can be found in the [56], [57]. It has to be noted that all considered algorithms use their default parameter settings [35], [54], [55], whilst the proposed SPAM-AOS utilizes its default parameter values as previously described.

Throughout this section, we adopt the experimental protocol used in [35]. Specifically, for each algorithm and each benchmark function, we conduct 100 independent runs. For each run, a budget of $maxFES = 5000 \cdot D$ function evaluations is employed, where D is the dimensionality of the problem. Tables I, II, III, and IV, show the results of our experiments in terms of final average error and corresponding standard deviation. A **boldface** font has been used to indicate the best performing algorithm, for each problem. To assess the statistical significance of the observed performance differences, for each problem and each algorithm we apply the non-parametric Wilcoxon rank sum test [58] between the corresponding algorithm and the proposed SPAM-AOS. The mark "+" denotes the cases where the null hypothesis is rejected at the 5% significance level and SPAM-AOS performs significantly better. Similarly, the mark "-" shows the rejection of the null hypothesis at the 5% level of significance and SPAM-AOS exhibits inferior performance. In the remaining cases, which we mark with "=", the null hypothesis is accepted which indicates that the performance of the algorithms being compared is statistically indistinguishable.

To this end, Tables I, II, III, and IV exhibit the extensive experimental results of all algorithms on the 10, 30, 50-dimensional CEC 2013 and on the 100-dimensional BBOB 2010 benchmark sets respectively. It can be clearly observed that, in the majority of the cases, SPAM-AOS either significantly outperforms the other algorithms or it behaves equally well. There are some cases where it produces an inferior performance (10-dimensional cases against the MDE-pBX algorithm), but this behavior radically changes as the dimensionality increases. Specifically, the impact of the adaptive operator selection approach on SPAM-AOS is evident since for the majority of the cases the performance gains are significantly better than the original SPAM algorithm. Similarly, comparing SPAM-AOS with CCPSO2, we can observe that SPAM-AOS exhibits superior performance in more than 60% of the tested cases. MDE-pBX is better than SPAM-AOS in the majority of the 10-dimensional CEC 2013 functions, but once more this behavior radically changes in the remaining experiments.

Finally, to have a general statistical sense of the significance of the algorithms across all problems and to alleviate the problem of having Type I errors in multiple comparisons with a higher probability, we employ the Holm-Bonferroni correction, see [59]. As such, Table V reports the rankings of the algorithm and the numerical results from the Holm-Bonferroni test. Clearly, the performance differences between SPAM-AOS and the other implemented algorithms are statistically significant.

TABLE V. Holm-Bonferroni procedure (reference: SPAM-AOS, Rank = 2.94e+00)

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	MDE-pBX	2.56e+00	-2.86e+00	2.13e-03	5.00e-02	Rejected
2	SPAM	2.25e+00	-5.10e+00	1.67e-07	2.50e-02	Rejected
3	CCPSO2	2.18e+00	-5.65e+00	8.14e-09	1.67e-02	Rejected

To summarize the performance obtained by the implemented algorithms, we provide a graphical illustration of their overall performance across all benchmark functions used in this suite of experiments. As such, we employ the Empirical Cumulative Distribution Function (ECDF) graph of the normalized performance. In detail, to be able to compare all algorithms on different benchmark problems we normalize their performance values per benchmark problem linearly in a common range, such as in $[0, 1]$. Note that, in the current setting, performance is measured by the optimal objective value found by an algorithm on one execution run. Thus, normalized performance values close to zero indicate best performance, whilst the performance becomes worse as the values increase to one. Having calculated the normalized performance values of an algorithm $\mathcal{A}$ over all benchmark functions, ECDF values can be measured according to $ECDF(x) = \frac{1}{n}\sum_{i=1}^{n} I_{(-\infty,x]}(x_i)$, where n is the number of benchmark functions times the number of independent executions of algorithm $\mathcal{A}$ per benchmark function and $I_{(-\infty,x]}(\cdot)$ is the indicator function which is equal to one if $x_i \leq x$, and to zero otherwise. Intuitively, an ECDF curve of an algorithm $\mathcal{A}$ depicts the empirical probability of observing a performance value y that is less than or equal to y. This enables a summarizing comparison between different algorithms, since higher ECDF values for the same normalized performance value indicate better performance.

Figure 1 illustrates the ECDF graph of the four implemented algorithms across all benchmark sets considered in this paper. It can be clearly observed that the SPAM-AOS algorithm exhibits great potential across all benchmark functions, since for the same performance value it always has higher ECDF values against the other algorithms. As such, it is more likely that it can provide better performance gains against the other three algorithms. Regarding the overall performance of the remaining algorithms, SPAM comes second since, for the lower normalized performance values, it exhibits higher cumulative frequencies against CCPSO2 and MDE-pBX. Similarly, MDE-pBX outperforms CCPSO2, since its ECDF curve values are always higher than the corresponding ECDF values of the CCPSO2 algorithm.

V. Conclusion

This paper proposes the integration of a technique normally used within hyper-heuristic frameworks within an MC

TABLE I. AVERAGE ERROR ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REF.: SPAM-AOS) ON CEC2013 [56] IN 10 DIMENSIONS.

	SPAM-AOS	SPAM		CCPSO2		MDE-pBX	
f_1	**0.00e+00 ± 0.00e+00**	0.00e+00 ± 0.00e+00	=	3.08e-03 ± 1.05e-02	+	0.00e+00 ± 2.27e-14	=
f_2	0.00e+00 ± 1.36e-13	**0.00e+00 ± 0.00e+00**	=	1.80e+06 ± 1.21e+06	+	2.54e+03 ± 5.07e+03	+
f_3	**1.31e+00 ± 3.50e+00**	1.08e+02 ± 7.77e+02	=	7.41e+07 ± 1.12e+08	+	1.41e+05 ± 1.23e+06	+
f_4	**0.00e+00 ± 0.00e+00**	0.00e+00 ± 0.00e+00	=	1.05e+04 ± 2.69e+03	+	3.82e+00 ± 3.15e+01	+
f_5	1.14e-13 ± 6.63e-14	3.46e-10 ± 1.34e-09	+	2.20e-02 ± 6.13e-02	+	**0.00e+00 ± 7.01e-14**	-
f_6	**4.11e+00 ± 4.78e+00**	5.63e+00 ± 4.78e+00	+	4.67e+00 ± 7.85e+00	+	5.70e+00 ± 4.83e+00	=
f_7	6.04e+01 ± 6.04e+01	7.57e+10 ± 7.18e+11	=	3.99e+01 ± 1.26e+01	=	**7.37e+00 ± 1.02e+01**	-
f_8	**2.04e+01 ± 1.38e-01**	2.05e+01 ± 1.22e-01	+	2.04e+01 ± 7.48e-02	+	2.05e+01 ± 9.69e-02	+
f_9	6.73e+00 ± 1.72e+00	1.42e+01 ± 4.69e+00	+	5.48e+00 ± 8.99e-01	-	**2.16e+00 ± 1.39e+00**	-
f_{10}	1.48e-02 ± 1.40e-02	**1.47e-02 ± 1.40e-02**	=	1.93e+00 ± 9.27e-01	+	1.06e-01 ± 8.03e-02	+
f_{11}	6.30e+00 ± 2.86e+00	1.67e+01 ± 7.81e+01	=	**2.76e+00 ± 1.85e+00**	-	2.89e+00 ± 1.72e+00	-
f_{12}	1.80e+01 ± 9.61e+00	1.27e+02 ± 2.01e+02	+	3.39e+01 ± 1.02e+01	+	**1.02e+01 ± 4.53e+00**	-
f_{13}	3.55e+01 ± 1.52e+01	2.83e+02 ± 4.49e+02	+	4.22e+01 ± 8.88e+00	+	**1.94e+01 ± 8.85e+00**	-
f_{14}	2.18e+02 ± 1.05e+02	7.63e+02 ± 6.39e+02	+	**8.67e+01 ± 6.15e+01**	-	1.08e+02 ± 9.77e+01	-
f_{15}	1.00e+03 ± 3.65e+02	1.64e+03 ± 4.51e+02	+	1.03e+03 ± 2.70e+02	=	**7.56e+02 ± 2.63e+02**	-
f_{16}	**2.92e-01 ± 1.98e-01**	5.32e-01 ± 6.39e-01	+	1.31e+00 ± 2.35e-01	+	5.74e-01 ± 4.62e-01	+
f_{17}	1.58e+01 ± 4.28e+00	2.19e+02 ± 4.34e+02	+	1.79e+01 ± 2.64e+00	+	**1.32e+01 ± 1.92e+00**	-
f_{18}	4.17e+01 ± 1.42e+01	7.78e+02 ± 5.07e+02	+	5.82e+01 ± 6.30e+00	+	**2.02e+01 ± 5.18e+00**	-
f_{19}	7.33e-01 ± 3.44e-01	9.20e-01 ± 3.30e-01	+	1.00e+00 ± 3.69e-01	+	**6.57e-01 ± 2.22e-01**	=
f_{20}	4.11e+00 ± 3.64e-01	4.45e+00 ± 4.82e-01	+	3.59e+00 ± 2.16e-01	-	**2.73e+00 ± 6.04e-01**	-
f_{21}	**2.86e+02 ± 1.26e+02**	3.25e+02 ± 1.14e+02	+	3.68e+02 ± 6.68e+01	+	3.98e+02 ± 1.99e+01	+
f_{22}	3.34e+02 ± 1.19e+02	1.19e+03 ± 8.95e+02	+	**1.23e+02 ± 6.60e+01**	-	1.77e+02 ± 1.37e+02	-
f_{23}	1.52e+03 ± 4.16e+02	2.23e+03 ± 5.07e+02	+	1.37e+03 ± 2.82e+02	-	**8.43e+02 ± 3.48e+02**	-
f_{24}	**1.93e+02 ± 4.46e+01**	2.72e+02 ± 1.43e+02	+	2.11e+02 ± 1.80e+01	=	2.05e+02 ± 5.21e+00	+
f_{25}	2.15e+02 ± 1.90e+01	2.52e+02 ± 7.08e+01	+	2.12e+02 ± 1.46e+01	-	**2.01e+02 ± 8.24e+00**	-
f_{26}	1.67e+02 ± 6.02e+01	2.51e+02 ± 1.45e+02	+	1.71e+02 ± 2.37e+01	+	**1.40e+02 ± 4.16e+01**	-
f_{27}	3.81e+02 ± 7.32e+01	4.24e+02 ± 2.72e+02	=	4.33e+02 ± 5.71e+01	+	**3.04e+02 ± 1.72e+01**	-
f_{28}	**2.92e+02 ± 9.79e+01**	1.00e+03 ± 1.14e+03	+	4.01e+02 ± 1.63e+02	+	3.04e+02 ± 5.53e+01	+

TABLE II. AVERAGE ERROR ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REF.: SPAM-AOS) ON CEC2013 [56] IN 30 DIMENSIONS.

	SPAM-AOS	SPAM		CCPSO2		MDE-pBX	
f_1	0.00e+00 ± 2.05e-13	**0.00e+00 ± 2.01e-13**	=	1.36e-12 ± 6.01e-12	+	2.27e-13 ± 4.86e-13	+
f_2	1.75e+03 ± 1.90e+03	**1.58e+03 ± 1.50e+03**	=	2.14e+06 ± 1.04e+06	+	2.70e+05 ± 2.62e+05	+
f_3	**7.88e+05 ± 1.79e+06**	9.96e+05 ± 1.94e+06	=	1.13e+09 ± 1.18e+09	+	5.19e+07 ± 1.18e+08	+
f_4	**5.16e-04 ± 4.54e-03**	4.05e-02 ± 4.03e-01	=	5.64e+04 ± 2.09e+04	+	3.49e+02 ± 3.18e+02	+
f_5	**1.14e-13 ± 6.49e-13**	1.09e-07 ± 1.00e-06	+	3.04e-07 ± 8.74e-07	+	1.09e-10 ± 1.00e-09	+
f_6	**9.73e-02 ± 4.74e-01**	4.80e-01 ± 2.74e+00	=	3.44e+01 ± 2.78e+01	+	3.41e+01 ± 2.77e+01	+
f_7	**4.00e+01 ± 2.58e+01**	7.54e+04 ± 7.50e+05	=	1.19e+02 ± 2.33e+01	+	5.61e+01 ± 1.90e+01	+
f_8	**2.09e+01 ± 7.13e-02**	2.10e+01 ± 5.45e-02	+	2.10e+01 ± 5.44e-02	+	2.10e+01 ± 5.93e-02	+
f_9	3.00e+01 ± 3.58e+00	3.16e+01 ± 5.34e+00	+	3.02e+01 ± 2.20e+00	=	**2.16e+01 ± 4.36e+00**	-
f_{10}	1.28e-02 ± 7.93e-03	**1.01e-02 ± 5.30e-03**	-	2.00e-01 ± 9.45e-02	+	1.81e-01 ± 1.10e-01	+
f_{11}	2.94e+01 ± 6.43e+00	2.50e+01 ± 6.22e+00	-	**5.76e-01 ± 6.49e-01**	-	4.68e+01 ± 1.54e+01	+
f_{12}	9.80e+01 ± 6.27e+01	3.34e+02 ± 6.51e+02	=	2.13e+02 ± 5.62e+01	+	**6.91e+01 ± 2.20e+01**	=
f_{13}	1.99e+02 ± 6.82e+01	5.18e+02 ± 9.98e+02	=	2.58e+02 ± 4.39e+01	+	**1.50e+02 ± 3.56e+01**	-
f_{14}	7.83e+02 ± 2.03e+02	1.85e+03 ± 1.67e+03	+	**6.57e+00 ± 3.69e+00**	-	1.20e+03 ± 4.25e+02	+
f_{15}	4.78e+03 ± 7.59e+02	4.63e+03 ± 8.62e+02	=	4.03e+03 ± 4.77e+02	-	**4.01e+03 ± 7.00e+02**	-
f_{16}	1.42e-01 ± 1.23e-01	**1.26e-01 ± 6.29e-02**	=	2.40e+00 ± 4.03e-01	+	1.32e+00 ± 8.61e-01	+
f_{17}	5.71e+01 ± 7.70e+00	2.69e+02 ± 8.25e+02	+	**3.13e+01 ± 4.89e-01**	-	6.89e+01 ± 1.24e+01	+
f_{18}	2.37e+02 ± 5.44e+01	8.63e+02 ± 1.37e+03	+	2.44e+02 ± 5.78e+01	=	**8.31e+01 ± 1.66e+01**	-
f_{19}	2.64e+00 ± 6.63e-01	2.76e+00 ± 8.35e-01	=	**8.55e-01 ± 1.71e-01**	-	9.10e+00 ± 4.94e+00	+
f_{20}	1.45e+01 ± 5.14e-01	1.46e+01 ± 4.86e-01	+	1.39e+01 ± 4.52e-01	-	**1.09e+01 ± 7.97e-01**	-
f_{21}	2.40e+02 ± 5.62e+01	**2.35e+02 ± 5.54e+01**	=	2.58e+02 ± 7.21e+01	+	3.09e+02 ± 7.63e+01	+
f_{22}	1.08e+03 ± 3.02e+02	2.37e+03 ± 2.02e+03	+	**1.21e+02 ± 7.28e+01**	-	1.11e+03 ± 5.46e+02	=
f_{23}	6.05e+03 ± 9.37e+02	5.95e+03 ± 1.04e+03	=	5.26e+03 ± 7.22e+02	-	**4.47e+03 ± 7.32e+02**	-
f_{24}	3.00e+02 ± 1.88e+02	3.20e+02 ± 2.59e+02	=	2.81e+02 ± 1.08e+01	-	**2.31e+02 ± 1.11e+01**	-
f_{25}	2.94e+02 ± 1.84e+01	2.95e+02 ± 1.75e+01	=	3.03e+02 ± 6.25e+00	+	**2.75e+02 ± 1.55e+01**	-
f_{26}	2.80e+02 ± 8.66e+01	3.07e+02 ± 2.44e+02	=	**2.02e+02 ± 4.53e+00**	=	2.16e+02 ± 4.31e+01	-
f_{27}	8.27e+02 ± 1.92e+02	8.63e+02 ± 2.08e+02	=	1.07e+03 ± 1.13e+02	+	**6.55e+02 ± 1.13e+02**	-
f_{28}	6.27e+02 ± 1.36e+03	1.04e+03 ± 2.30e+03	+	5.43e+02 ± 5.77e+02	-	**3.11e+02 ± 1.11e+02**	-

TABLE III. AVERAGE ERROR ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REF.: SPAM-AOS) ON CEC2013 [56] IN 50 DIMENSIONS.

	SPAM-AOS	SPAM		CCPSO2		MDE-pBX	
f_1	**2.27e-13 ± 0.00e+00**	2.27e-13 ± 0.00e+00	=	7.05e-12 ± 3.53e-11	+	3.32e-11 ± 2.60e-10	+
f_2	2.66e+04 ± 1.35e+04	**2.64e+04 ± 1.24e+04**	=	4.37e+06 ± 2.29e+06	+	9.06e+05 ± 4.90e+05	+
f_3	8.55e+06 ± 1.43e+07	**6.32e+06 ± 1.11e+07**	=	3.09e+09 ± 3.03e+09	+	1.42e+08 ± 1.57e+08	+
f_4	5.59e+02 ± 5.01e+02	**4.79e+02 ± 6.48e+02**	-	1.08e+05 ± 3.86e+04	+	1.09e+03 ± 8.33e+02	+
f_5	**3.41e-13 ± 1.05e-12**	3.81e-10 ± 6.24e-10	+	3.92e-04 ± 3.89e-03	+	2.54e-05 ± 2.52e-04	=
f_6	**2.74e+01 ± 1.75e+01**	3.96e+01 ± 1.34e+01	+	4.74e+01 ± 1.34e+01	+	5.67e+01 ± 2.24e+01	+
f_7	**4.75e+01 ± 2.38e+01**	4.83e+01 ± 1.97e+01	=	1.43e+02 ± 2.39e+01	+	6.81e+01 ± 1.22e+01	+
f_8	2.11e+01 ± 6.66e-02	**2.11e+01 ± 6.58e-02**	=	2.12e+01 ± 3.86e-02	+	2.12e+01 ± 4.36e-02	+
f_9	5.49e+01 ± 5.03e+00	6.85e+01 ± 1.12e+01	+	5.87e+01 ± 3.26e+00	+	**4.27e+01 ± 6.99e+00**	-
f_{10}	**1.24e-02 ± 7.26e-03**	1.28e-02 ± 8.01e-03	=	2.03e-01 ± 1.80e-01	+	4.09e-01 ± 5.57e-01	+
f_{11}	5.87e+01 ± 1.05e+01	5.13e+01 ± 8.62e+00	-	**9.07e-01 ± 8.53e-01**	-	1.21e+02 ± 2.97e+01	+
f_{12}	2.98e+02 ± 1.43e+02	3.18e+02 ± 2.59e+02	=	4.55e+02 ± 8.03e+01	+	**1.62e+02 ± 3.45e+01**	-
f_{13}	5.33e+02 ± 1.12e+02	5.61e+02 ± 2.38e+02	=	5.69e+02 ± 8.18e+01	+	**3.22e+02 ± 5.39e+01**	-
f_{14}	1.41e+03 ± 3.09e+02	3.09e+03 ± 2.95e+03	+	**7.35e+00 ± 3.55e+00**	-	2.79e+03 ± 8.06e+02	+
f_{15}	8.50e+03 ± 1.09e+03	8.54e+03 ± 1.05e+03	=	8.31e+03 ± 8.71e+02	=	**7.58e+03 ± 8.01e+02**	-
f_{16}	**8.36e-02 ± 3.88e-02**	9.21e-02 ± 4.43e-02	=	2.75e+00 ± 5.96e-01	+	1.93e+00 ± 8.76e-01	+
f_{17}	9.62e+01 ± 1.11e+01	9.83e+01 ± 7.83e+00	=	**5.16e+01 ± 3.28e-01**	-	1.79e+02 ± 3.56e+01	+
f_{18}	5.37e+02 ± 9.93e+01	1.86e+03 ± 2.49e+03	+	4.87e+02 ± 9.77e+01	-	**1.86e+02 ± 3.17e+01**	-
f_{19}	4.73e+00 ± 8.92e-01	4.99e+00 ± 1.09e+00	=	**1.49e+00 ± 2.32e-01**	-	3.94e+01 ± 2.10e+01	+
f_{20}	2.44e+01 ± 2.86e-01	2.44e+01 ± 4.10e-01	=	2.33e+01 ± 8.19e-01	-	**2.01e+01 ± 9.17e-01**	-
f_{21}	**4.27e+02 ± 3.30e+02**	5.00e+02 ± 3.85e+02	=	4.42e+02 ± 3.45e+02	+	8.91e+02 ± 3.44e+02	+
f_{22}	2.06e+03 ± 3.17e+02	3.99e+03 ± 3.56e+03	+	**1.11e+02 ± 9.60e+01**	-	3.22e+03 ± 1.06e+03	+
f_{23}	1.12e+04 ± 1.27e+03	1.16e+04 ± 1.25e+03	+	1.09e+04 ± 1.34e+03	=	**9.08e+03 ± 1.05e+03**	-
f_{24}	3.62e+02 ± 2.14e+02	1.20e+03 ± 1.04e+03	+	3.60e+02 ± 9.64e+00	-	**2.88e+02 ± 1.56e+01**	-
f_{25}	3.79e+02 ± 1.99e+01	4.51e+02 ± 1.27e+02	+	3.97e+02 ± 1.08e+01	+	**3.68e+02 ± 1.48e+01**	-
f_{26}	3.08e+02 ± 2.97e+02	5.15e+02 ± 6.42e+02	+	**2.15e+02 ± 4.95e+01**	-	3.55e+02 ± 7.46e+01	+
f_{27}	1.28e+03 ± 2.33e+02	1.32e+03 ± 3.32e+02	=	1.82e+03 ± 8.56e+01	+	**1.23e+03 ± 1.49e+02**	=
f_{28}	1.63e+03 ± 2.22e+03	3.40e+03 ± 5.33e+03	=	7.24e+02 ± 1.08e+03	-	**5.05e+02 ± 5.99e+02**	=

algorithm for continuous optimization. The employed algorithmic structure and operators are the same as those used by a MC approach previously proposed in literature. The hyper-heuristic adaptive technique integrates, on the top of the original memetic logic, a success-based adaptation. The resulting framework appears to improve upon the original MC approach and is competitive with modern meta-heuristics. This study can be seen as an attempt to consider about algorithmic design from a metaphor-free perspective and, in the specific case, to show how some areas of MC and hyper-heuristics contain essentially very similar ideas (if not even the same idea).

TABLE IV. AVERAGE ERROR ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REF.: SPAM-AOS) ON BBOB2010 [57] IN 100 DIMENSIONS.

	SPAM-AOS	SPAM		CCPSO2		MDE-pBX	
f_1	$\mathbf{2.42e-13 \pm 2.12e-13}$	$2.42e-13 \pm 2.13e-13$	=	$4.12e-13 \pm 1.97e-13$	+	$8.12e-07 \pm 3.91e-06$	+
f_2	$\mathbf{1.71e-13 \pm 1.55e-13}$	$2.84e-13 \pm 1.92e-13$	+	$9.66e-13 \pm 1.77e-12$	+	$3.22e-02 \pm 2.86e-01$	+
f_3	$1.04e+02 \pm 1.70e+01$	$9.69e+01 \pm 1.51e+01$	-	$\mathbf{8.09e+00 \pm 8.40e+00}$	-	$5.06e+02 \pm 9.57e+01$	+
f_4	$1.37e+02 \pm 2.01e+01$	$1.34e+02 \pm 1.86e+01$	=	$\mathbf{2.24e+01 \pm 1.31e+01}$	-	$8.32e+02 \pm 1.28e+02$	+
f_5	$4.93e-08 \pm 2.91e-07$	$\mathbf{2.13e-11 \pm 8.26e-12}$	-	$2.41e-04 \pm 1.20e-03$	+	$7.53e+00 \pm 1.01e+01$	+
f_6	$6.15e-08 \pm 7.12e-08$	$\mathbf{1.79e-11 \pm 1.14e-11}$	-	$8.94e+01 \pm 4.02e+01$	+	$3.36e+01 \pm 2.62e+01$	+
f_7	$\mathbf{5.26e+01 \pm 1.40e+01}$	$5.32e+01 \pm 1.36e+01$	=	$3.45e+02 \pm 4.90e+01$	+	$2.79e+02 \pm 7.50e+01$	+
f_8	$3.66e+01 \pm 1.11e+01$	$\mathbf{3.49e+01 \pm 9.88e+00}$	-	$1.20e+02 \pm 3.42e+01$	+	$1.78e+02 \pm 6.58e+01$	+
f_9	$\mathbf{4.39e+01 \pm 7.26e+00}$	$4.43e+01 \pm 6.96e+00$	+	$1.06e+02 \pm 2.78e+01$	+	$1.29e+02 \pm 3.80e+01$	+
f_{10}	$5.27e+02 \pm 1.50e+02$	$\mathbf{4.83e+02 \pm 1.43e+02}$	=	$2.62e+04 \pm 6.64e+03$	+	$1.51e+04 \pm 6.24e+03$	+
f_{11}	$8.72e+01 \pm 3.08e+01$	$7.86e+01 \pm 2.53e+01$	=	$5.45e+02 \pm 1.95e+02$	+	$\mathbf{1.62e+01 \pm 6.78e+00}$	-
f_{12}	$5.29e-02 \pm 1.98e-01$	$\mathbf{2.61e-02 \pm 8.27e-02}$	=	$7.95e+00 \pm 1.20e+01$	+	$2.70e+01 \pm 1.29e+02$	+
f_{13}	$\mathbf{1.06e+00 \pm 1.24e+00}$	$1.31e+00 \pm 1.78e+00$	=	$3.16e+00 \pm 4.20e+00$	+	$4.80e+00 \pm 9.83e+00$	+
f_{14}	$5.07e-05 \pm 6.32e-06$	$\mathbf{3.17e-05 \pm 3.66e-06}$	-	$1.32e-03 \pm 2.34e-04$	+	$2.21e-03 \pm 1.70e-03$	+
f_{15}	$2.75e+02 \pm 4.48e+01$	$\mathbf{2.73e+02 \pm 4.25e+01}$	=	$1.33e+03 \pm 2.32e+02$	+	$6.53e+02 \pm 9.63e+01$	+
f_{16}	$\mathbf{2.31e+00 \pm 8.27e-01}$	$2.42e+00 \pm 7.82e-01$	=	$2.74e+01 \pm 4.27e+00$	+	$1.35e+01 \pm 3.23e+00$	+
f_{17}	$8.55e+00 \pm 4.71e+00$	$8.59e+00 \pm 4.52e+00$	=	$8.65e+00 \pm 1.62e+00$	=	$\mathbf{3.36e+00 \pm 4.02e-01}$	-
f_{18}	$1.84e+01 \pm 1.15e+01$	$1.95e+01 \pm 1.32e+01$	=	$3.30e+01 \pm 6.61e+00$	+	$\mathbf{1.19e+01 \pm 2.08e+00}$	-
f_{19}	$1.96e+00 \pm 4.27e-01$	$\mathbf{1.67e+00 \pm 2.95e-01}$	-	$7.90e+00 \pm 1.29e+00$	+	$2.26e+00 \pm 6.47e-01$	+
f_{20}	$1.14e+00 \pm 1.09e-01$	$1.25e+00 \pm 1.66e-01$	+	$\mathbf{4.95e-01 \pm 6.73e-02}$	-	$2.12e+00 \pm 9.26e-02$	+
f_{21}	$3.89e+00 \pm 4.11e+00$	$4.32e+00 \pm 6.61e+00$	=	$\mathbf{3.36e+00 \pm 3.38e+00}$	=	$3.79e+00 \pm 5.07e+00$	=
f_{22}	$7.94e+00 \pm 9.28e+00$	$\mathbf{4.93e+00 \pm 6.86e+00}$	=	$5.11e+00 \pm 5.83e+00$	-	$8.94e+00 \pm 8.84e+00$	+
f_{23}	$8.23e-01 \pm 3.93e-01$	$\mathbf{7.60e-01 \pm 3.66e-01}$	=	$2.52e+00 \pm 4.22e-01$	+	$2.48e+00 \pm 9.31e-01$	+
f_{24}	$\mathbf{3.10e+02 \pm 6.43e+01}$	$3.19e+02 \pm 5.82e+01$	=	$1.08e+03 \pm 1.53e+02$	+	$3.48e+02 \pm 5.00e+01$	+

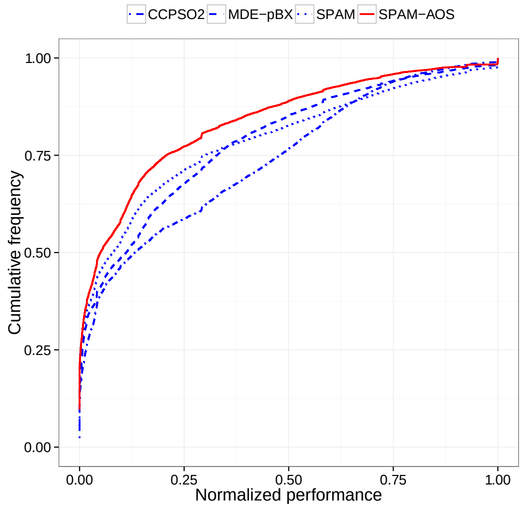

Fig. 1. The Empirical Cumulative Distribution Function graph of the normalized objective values for all algorithms, across all problem instances and runs. It can be clearly observed that SPAM-AOS always achieves higher cumulative frequency of lower objective values across all problem instances and runs (better performance gains).

ACKNOWLEDGMENTS

E.K. Burke and M.G. Epitropakis would like to thank EPSRC for their support for this work through grant EP/J017515/1. This research is also supported by the Academy of Finland, Akatemiatutkija 130600, "Algorithmic design issues in Memetic Computing".

REFERENCES

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[2] A. Auger and O. Teytaud, "Continuous lunches are free!" in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 916–922.

[3] R. Poli and M. Graff, "There is a free lunch for hyper-heuristics, genetic programming and computer scientists," in *EuroGP*, 2009, pp. 195–207.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley Publishing Co., 1989.

[5] J. Du and R. Rada, "Memetic algorithms, domain knowledge, and financial investing," *Memetic Computing*, vol. 4, no. 2, pp. 109–125, 2012.

[6] J. Smith, "The co-evolution of memetic algorithms for protein structure prediction," in *Recent Advances in Memetic Algorithms*, ser. Studies in Fuzziness and Soft Computing, W. Hart, N. Krasnogor, and J. Smith, Eds. Springer, 2004, vol. 166, pp. 105–128.

[7] F. Neri and E. Mininno, "Memetic Compact Differential Evolution for Cartesian Robot Control," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 54–65, 2010.

[8] F. Neri, C. Cotta, and P. Moscato, *Handbook of Memetic Algorithms*, ser. Studies in Computational Intelligence. Springer, 2011, vol. 379.

[9] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[10] R. Mallipeddi, S. Mallipeddi, and P. N. Suganthan, "Ensemble strategies with adaptive evolutionary programming," *Information Sciences*, vol. 180, no. 9, pp. 1571–1581, 2010.

[11] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic Approach to Scheduling a Sales Summit," in *Proceedings of the Third International Conference on Practice and Theory of Automated Timetabling*, ser. LNCS. Springer, 2000, vol. 2079, pp. 176–190.

[12] E. K. Burke, G. Kendall, and E. Soubeiga, "A Tabu Search hyperheuristic for Timetabling and Rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[13] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyperheuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, pp. 177–192, 2007.

[14] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[15] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of Meta-heuristics*, ser. International Series in Operations Research & Management Science, M. Gendreau and J.-Y. Potvin, Eds. Springer US, 2010, no. 146, pp. 449–468.

[16] E. K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science, F. Glover and G. A. Kochenberger, Eds. Springer US, 2003, no. 57, pp. 457–474.

[17] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[18] A. Fialho, "Adaptive operator selection for optimization," Ph.D. dissertation, Université Paris-Sud XI, Orsay, France, 2010.

[19] A. Fialho, L. D. Costa, M. Schoenauer, and M. Sebag, "Analyzing

bandit-based adaptive operator selection mechanisms," *Annals of Mathematics and Artificial Intelligence*, vol. 60, no. 1-2, pp. 25–64, 2010.

[20] J. Maturana, . Fialho, F. Saubion, M. Schoenauer, F. Lardeux, and M. Sebag, "Adaptive operator selection and management in evolutionary algorithms," in *Autonomous Search*, Y. Hamadi, E. Monfroy, and F. Saubion, Eds. Springer Berlin Heidelberg, 2012, pp. 161–189.

[21] K. A. Dowsland, E. Soubeiga, and E. Burke, "A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation," *European Journal of Operational Research*, vol. 179, no. 3, pp. 759–774, 2007.

[22] E. Özcan, M. Misir, G. Ochoa, and E. K. Burke, "A reinforcement learning-great-deluge hyper-heuristic for examination timetabling," *International Journal of Applied Metaheuristic Computing*, vol. 1, no. 1, pp. 39–59, 2010.

[23] G. Acampora, M. Gaeta, and V. Loia, "Hierarchical optimization of personalized experiences for e-learning systems through evolutionary models," *Neural Computing and Applications*, vol. 20, no. 5, pp. 641–657, 2011.

[24] G. Acampora, J. M. Cadenas, V. Loia, and E. M. Ballester, "A multi-agent memetic system for human-based knowledge selection," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 41, no. 5, pp. 946–960, 2011.

[25] G. Ochoa, M. Hyde, T. Curtois, J. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. Parkes, S. Petrovic, and E. K. Burke, "Hyflex: A benchmark framework for cross-domain heuristic search," in *Evolutionary Computation in Combinatorial Optimization*, ser. LNCS, J.-K. Hao and M. Middendorf, Eds. Springer Berlin Heidelberg, 2012, vol. 7245, pp. 136–147.

[26] M. Misir, K. Verbeeck, P. D. Causmaecker, and G. V. Berghe, "An intelligent hyper-heuristic framework for CHeSC 2011," in *Learning and Intelligent Optimization*, ser. LNCS, Y. Hamadi and M. Schoenauer, Eds. Springer Berlin Heidelberg, 2012, pp. 461–466.

[27] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 243–259, 2009.

[28] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-Based Algorithm Portfolios for Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 782–800, 2010.

[29] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown, "SATzilla: Portfolio-based algorithm selection for SAT," *Journal of Artificial Intelligence Research*, vol. 32, pp. 565–606, 2008.

[30] H. H. Hoos, "Programming by optimization," *Commun. ACM*, vol. 55, no. 2, pp. 70–80, 2012.

[31] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Tradeoffs in the empirical evaluation of competing algorithm designs," *Ann. Math. Artif. Intell.*, vol. 60, no. 1-2, pp. 65–89, 2010.

[32] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, "Algorithm runtime prediction: Methods & evaluation," *Artificial Intelligence*, vol. 206, pp. 79–111, 2014.

[33] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[34] F. Neri, E. Mininno, and G. Iacca, "Compact particle swarm optimization," *Information Sciences*, vol. 239, pp. 96–121, 2013.

[35] F. Caraffini, F. Neri, and L. Picinali, "An analysis on separability for memetic computing automatic design," *Information Sciences*, vol. 265, pp. 1–22, 2014.

[36] Y. S. Ong and A. J. Keane, "Meta-Lamarkian Learning in Memetic Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99–110, 2004.

[37] J. E. Smith, "Coevolving Memetic Algorithms: A Review and Progress Report," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 1, pp. 6–17, 2007.

[38] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[39] ——, "Memetic Algorithms: The Polynomial Local Search Complexity Theory Perspective," *J. Math. Model. Algorithms*, vol. 7, no. 1, pp. 3–24, 2008.

[40] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives," *IEEE Transactions on System Man and Cybernetics-part B*, vol. 37, no. 1, pp. 28–41, 2007.

[41] F. Neri, V. Tirronen, T. Kärkkäinen, and T. Rossi, "Fitness diversity based adaptation in Multimeme Algorithms: A comparative study," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2007, pp. 2374–2381.

[42] F. Neri, J. I. Toivanen, G. L. Cascella, and Y. S. Ong, "An Adaptive Multimeme Algorithm for Designing HIV Multidrug Therapies," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 2, pp. 264–278, 2007.

[43] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim, "Ockham's Razor in Memetic Computing: Three Stage Optimal Memetic Exploration," *Information Sciences*, vol. 188, pp. 17–43, 2012.

[44] F. Caraffini, F. Neri, G. Iacca, and A. Mol, "Parallel memetic structures," *Information Sciences*, vol. 227, no. 0, pp. 60 – 82, 2013.

[45] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Tracking differential evolution algorithms: An adaptive approach through multinomial distribution tracking with exponential forgetting," in *Artificial Intelligence: Theories and Applications*, ser. LNCS, Maglogiannis, Plagianakos, and Vlahavas, Eds. Springer, 2012, no. 7297, pp. 214–222.

[46] M. Epitropakis, D. Tasoulis, N. Pavlidis, V. Plagianakos, and M. Vrahatis, "Tracking particle swarm optimizers: An adaptive approach through multinomial distribution tracking with exponential forgetting," in *IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1–8.

[47] D. Thierens, "An adaptive pursuit strategy for allocating operator probabilities," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005, p. 15391546.

[48] ——, "Adaptive strategies for operator allocation," in *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence, F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Springer Berlin Heidelberg, 2007, no. 54, pp. 77–90, 00042.

[49] L.-Y. Tseng and C. Chen, "Multiple trajectory search for Large Scale Global Optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2008, pp. 3052–3059.

[50] H. H. Rosenbrock, "An automatic Method for finding the greatest or least Value of a Function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.

[51] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory And Algorithms*. Wiley-Interscience, 2006.

[52] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[53] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998, 02767.

[54] X. Li and X. Yao, "Cooperatively Coevolving Particle Swarms for Large Scale Optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 2, pp. 210–224, 2012.

[55] S. Islam, S. Das, S. Ghosh, S. Roy, and P. Suganthan, "An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 482–500, 2012.

[56] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernndez-Daz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization," Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore, Tech. Rep. 201212, 2013.

[57] N. Hansen, A. Auger, S. Finck, R. Ros *et al.*, "Real-Parameter Black-Box Optimization Benchmarking 2010: Noiseless Functions Definitions," INRIA, Tech. Rep. RR-6829, 2010.

[58] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[59] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.