ICARUS: Identification of Complementary algoRithms by Uncovered Sets

Mario A. Muñoz
School of Mathematical Sciences
Monash University
Clayton, VIC Australia
e-mail: mario.munoz@monash.edu

Michael Kirley
Department of Computer and Information Systems
The University of Melbourne
Parkville, VIC Australia
e-mail: mkirley@unimelb.edu.au

***Abstract*—Since there is no single best performing algorithm for all problems, an algorithm portfolio would leverage the strengths of complementary algorithms to achieve the best performance. In this paper, we present and evaluate a new technique for designing algorithm portfolios for continuous black-box optimization problems, based on social choice and voting theory concepts. Our technique, which we call ICARUS, models the portfolio design task as an election, in which each problem 'votes' for a subset of preferred algorithms guided by a performance metric such as the number of fitness evaluations. The resulting 'uncovered set' of algorithms forms the portfolio. We demonstrate the efficacy of ICARUS using a suite of state-of-the-art evolutionary algorithms and benchmark continuous optimization problems. Our analysis confirms that ICARUS creates an algorithm portfolio where the expected performance is superior to a manually constructed portfolio.**

***Index Terms*—Algorithm portfolios, Black-box optimization, Continuous optimization, Heuristics**

I. Introduction

A significant challenge when attempting to solve a continuous black-box optimization (BBO) problem is to find a good quality solution as quickly as possible. Such problems often lack algebraic expressions, calculable derivatives and/or a well-defined structure. Consequently, the only reliable information that can be used to characterize BBO problems is the responses to a set of candidate solutions. Topologically, these problems can present large fluctuations in quality between adjacent solutions, local and global optima, and interdependencies between variables —features that are typically unknown. In addition, it is very common for such problems to involve computationally intensive simulations. Hence, finding a target solution is difficult.

The ubiquitous nature of BBO problems in scientific, engineering and practical applications [1], coupled with the challenge of attempting to solve these problems with limited resources and knowledge, has led to an increased number of reported algorithms over the last decades. An unexpected by-product of this algorithmic diversity is a compromised ability to master —or even be familiar with— all optimization algorithms [2]. This would not be an issue if all algorithms performed well on any problem. Unfortunately, this is not the case. Some algorithms are preferable to others for a given problem [3]–[5]. The diversity also eclipses significant algorithmic innovations [6] and fosters the recycling of ideas [7]–[10]. Furthermore, theoretical evaluation of several algorithms remains limited to simplified problems [11]. Therefore, selecting an appropriate algorithm for a given problem is at best cumbersome [12], even with expert knowledge on search algorithms, and skills in algorithm engineering and statistics [13].

One way to circumvent the 'algorithm selection' problem to some extent is to use an algorithm portfolio [12], [14], [15]. Algorithm portfolios try to combine the strength of individual algorithms that excel on various subsets of problem instances, when tackling a specific problem instance. Typically, component algorithms are run simultaneously, or in some cases, sequentially in order to find high quality solutions. However, the selection of the component algorithms poses a new set of challenges in itself.

Ideally, an algorithm portfolio should be composed of the smallest set of well-established and complementary algorithms. That is, sufficient evidence should be available indicating that the nominated algorithms can solve different types of problems. This implies that algorithm portfolio design can be formulated as a complex combinatorial optimization problem. For example, a portfolio could be generated by calculating the probability that it will be outperformed by any candidate algorithm [12]. Alternatively, a portfolio could be built incrementally by selecting the constituent algorithm with the lowest average rank on a set of problems, and then, add additional algorithms where ranks are uncorrelated to the portfolio [16]. Another approach consists of predicting the cost achieved by each algorithm after a number of evaluations, and pick the one with the least predicted cost [17].

In this paper, we take a different approach. We propose a simple heuristic method, which we call ICARUS, used to create an algorithm portfolio for BBO problems. Our method is based on 'uncovered sets' (also known as Landau or Fishburn sets), a concept derived from voting systems theory [18]–[20]. ICARUS models the portfolio design task as an election, in which each problem 'votes' for a subset of preferred algorithms. The uncovered set is the group of alternative algorithms each of which can defeat every other alternative in the algorithm pool either directly or indirectly, based on a given algorithm performance metric. A property

978-1-5090-0623-6/16/$31.00

of the uncovered set is that all of its elements are Pareto optimal; hence, it represents the most reasonable outcome in an election, even under tactical voting [20].

ICARUS is deterministic. However, we do not have a theoretical proof that ICARUS is optimal. To provide a proof-of-concept, we show that ICARUS is capable of generating a subset of algorithms, drawn from a pool of well-known evolutionary algorithms for BBO problems, where the expected performance of the algorithm portfolio is superior to a manually constructed portfolio across of a suite of benchmark problems.

The remainder of this paper is organized as follows: Section II describes the rationale behind our heuristic method and presents the ICARUS algorithm in detail. Section III describes the experimental validation procedure. The results using a suite of state-of-the-art evolutionary algorithms and benchmark problems are presented in Section IV. The paper concludes in Section V with a summary of our findings and a discussion of the limitations of this work.

II. Method

Before describing ICARUS in detail, it important to define our notation. Without loosing generality over maximization, we assume minimization throughout this paper.

The goal in a BBO problem is to minimize a cost function $f : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X} \subset \mathbb{R}^D$ is the input space, $\mathcal{Y} \subset \mathbb{R}$ is the output space, and $D \in \mathbb{N}^*$ is the dimensionality of the problem. A candidate solution $\mathbf{x} \in \mathcal{X}$ is a D-dimensional vector, and $y \in \mathcal{Y}$ is the candidate's cost. A target objective value, $y_t \in \mathcal{Y}$, defines the upper bound of a satisfactory minimization performance from an algorithm.

Let $\mathcal{F}$ be the space of all possible BBO problems, and $F \subset \mathcal{F}$ be a subset of training problems. Let $\mathcal{A}$ be the set of candidate algorithms, and $A \subset \mathcal{A}$ be the algorithm portfolio, which must be the smallest subset of algorithms that solves the largest group of problems with near best performance.

Assume that for all $f \in F$, we have recorded a performance metric ρ for all the algorithms $\alpha \in \mathcal{A}$. Each f has a preference, $\mathbf{t}_f$, formally defined as a linear ordering of algorithms, $\{\alpha_b \succ \ldots \succ \alpha_w\}$, where α_b is the most preferred algorithm and α_w is the least preferred algorithm. An algorithm α_i is preferred over α_j if $\rho(f, \alpha_i) < \rho(f, \alpha_j)$. Preferences are truncated to the K best algorithms that $\rho \ll \infty$. Hence, K represents the maximum number of valid preferences, and it is a user-selected parameter. The preferences form a family set $\mathbf{T} = \{\mathbf{t}_f : f \in F\}$.

Algorithm 1 lists the pseudocode for our implementation of ICARUS. An inspection of the algorithm clearly indicates two key stages of the heuristic:

Preliminary stage:

In this stage, ICARUS generates a subset A_{BST}, which contains those algorithms considered the best for at least one training problem.

Iterative elimination stage:

In this stage, ICARUS finds the uncovered set of algorithms for all the unsolved problems, $\mathcal{U}$. The uncovered set is used in voting systems theory to identify the winners of an election [18]. Here the candidates are the algorithms, the voters are the problems, and the preferences are the ballots. Let α_i, α_j, and α_k be three candidate algorithms. Then α_i belongs to $\mathcal{U}$ if and only if it is preferred by the majority of problems than all other algorithms either directly or at one remove; i.e., for every α_j, either α_i is majority preferred to α_j, or there is a α_k such that α_i is majority preferred to α_k and α_k is majority preferred to α_j [20]. ICARUS adds all $\alpha \in \mathcal{U}$ to A. The iterative stage is repeated until all of $f \in F$ have at least one algorithm in A for which $\rho \ll \infty$.

ICARUS employs Petschel's[1] voting systems implementation to find $\mathcal{U}$. Our MATLAB implementation of ICARUS is available upon request.

III. Experimental Validation

To validate ICARUS it is necessary to nominate pools of problems and algorithms and evaluate the efficacy of the uncovered set voting protocol. In this section, we define a representative subset of optimization problems, F, a set of candidate algorithms $\mathcal{A}$, a performance metric, ρ, and describe how the algorithm portfolio is populated.

A. Optimization problems

We use the Comparing Continuous Optimizers (COCO) noiseless benchmark set [21] as F. The software implementation of COCO generates instances by translating and rotating the function in the input and output spaces. For example, let $f(\mathbf{x}) = \|\mathbf{R}(\mathbf{x} - \mathbf{x}_o)\|^2 + y_o$ be a functions in COCO, where $\mathbf{R}$ is an orthogonal rotation matrix, $\mathbf{x}_o$ and y_o cause translational shifts on the input and output space respectively. A function instance is generated by providing values for $\mathbf{R}$, $\mathbf{x}_o$, and y_o. To allow replicable experiments, the software uniquely identifies each instance using an index. For each function at each dimension, we analyze instances $[1, \ldots, 15]$ at $D = \{2, 5, 10, 20\}$. This gives us a total of 1440 problem instances for analysis.

B. Algorithms

We use the algorithms listed in Table I as $\mathcal{A}$. These algorithms were tested during the Black-Box Optimization Benchmarking (BBOB) sessions at the 2009 and 2010 GECCO conferences, and the results are publicly available at the benchmark website[2]. The results are stored as raw data files containing the numerical output of an algorithm run on a given problem. Hansen et al. in [21] provides a thorough description of the files' contents.

[1] http://www.mathworks.com.au/matlabcentral/fileexchange/28521-election

[2] http://coco.gforge.inria.fr/doku.php

Input: A set of algorithms, $\mathcal{A}$, a subset of problems F, a family set of algorithm preferences, $\mathbf{T}$, truncated to a number of valid preferences, K.
Output: An algorithm portfolio, $A \subset \mathcal{A}$.

$A = \emptyset$, $A_{\text{BST}} = \emptyset$, $F_{\text{USLV}} = F$, $q = |F_{\text{USLV}}|$;

// Preliminary stage

forall the $f \in F$ **do** $A_{\text{BST}} \leftarrow A_{\text{BST}} + \{\alpha_b : \alpha_b = \arg\min \rho(f, \alpha), \forall \alpha \in \mathcal{A}\}$; // Find $A_{\text{BST}} \subset \mathcal{A}$

// Iterative Elimination stage

while $q > 0$ **do** // While there are unsolved problems by $\alpha \in A$
  $\mathbf{B}_{|A_{\text{BST}}| \times |A_{\text{BST}}|} = \mathbf{0}$, $\mathbf{C}_{|A_{\text{BST}}| \times |A_{\text{BST}}|} = \mathbf{0}$, $\mathcal{U} = \emptyset$;
  foreach $f \in F_{USLV}$ **do** // For each unsolved problem
    foreach $\alpha_i \in A_{BST}$ **do** // Calculate the number of votes for each $\alpha \in A_{\text{BST}}$
      foreach $\alpha_j \in A_{BST}$ **do**
        if $(\alpha_i \succ \alpha_j) \wedge (\alpha_i \in \mathbf{t}_f)$ **then** $\mathbf{C}(i,j) \leftarrow \mathbf{C}(i,j) + 1$;
      end
    end
  end
  foreach $\alpha_i \in A_{BST}$ **do** // Check whether two algorithms have equal number of votes
    foreach $\alpha_j \in A_{BST}$ **do**
      if $\mathbf{C}(i,j) = \mathbf{C}(j,i)$ **then** $\mathbf{B}(i,j) = 1$;
    end
  end
  $\mathbf{B} \leftarrow (\mathbf{B} + \mathbf{BB}) > 1$;
  foreach $\alpha_i \in A_{BST}$ **do** // Check which $\alpha \in \mathcal{U}$
    flag $= 1$;
    foreach $\alpha_j \in A_{BST}$ **do**
      if $\mathbf{B}(i,j) \neq 1$ **then** flag $= 0$;
    end
    if *flag* $= 1$ **then** $\mathcal{U} \leftarrow \mathcal{U} + \{\alpha_i\}$; // If $\alpha_i \succ \alpha_j \forall \alpha_j \in A_{\text{BST}}$, then $\alpha_i \in \mathcal{U}$
  end
  foreach $f \in F_{USLV}$ **do** // Update F_{USLV}
    foreach $\alpha_i \in A$ **do**
      if $\rho(f, \alpha_i) \ll \infty$ **then** $F_{\text{USLV}} \leftarrow F_{\text{USLV}} - \{f\}$;
    end
  end
  $A \leftarrow A + \mathcal{U}$, $q = |F_{\text{USLV}}|$;
end

Algorithm 1: Identification of Complementary AlgoRithms by Uncovered Sets (ICARUS) method, which constructs the portfolio, A, which solve all the training problems, $f \in F$, with the near best performance. As convention, we used "$\leftarrow$" to imply assignment.

C. Performance metrics

The performance of an algorithm can be measured in terms of its robustness —how often is a quality solution found— or its efficiency —how many resources are needed to find a quality solution [22]. Robustness measures are useful when resources are limited [21], but they are harder to interpret as finding a m-times better solution is linked to the possible unknown problem difficulty [21]. On the other hand, efficiency measures use fixed solution qualities [22]. Therefore, they are preferable for comparing algorithms, because it is evident that an algorithm is superior if it reaches a target solution m-times faster than any other [21]. Hence, we focus on measuring the efficiency of an algorithm by using the expected running time, $\widehat{T}$, which estimates the average number of function evaluations required by an algorithm to reach y_t within a target precision for the first time [21]. A normalized, log-transformed version is calculated as follows:

TABLE I
SET OF ALGORITHMS FROM THE 2009 AND 2010 BBOB SESSIONS ($\mathcal{A}$). PERFORMANCE RESULTS FOR EACH ALGORITHM ARE AVAILABLE AT THE COCO WEBSITE.

$(1,2)$CMA-ES	CMA-ESPLUSSEL	MCS
$(1,2)^m$CMA-ES	CMAEGS	MOS
$(1,2)^m_s$CMA-ES	Cauchy-EDA	NBC-CMA
$(1,2)_s$CMA-ES	DE-F-AUC	Nelder-Doerr
$(1,4)$CMA-ES	DE-PSO	1/5th ES
$(1,4)^m$CMA-ES	DEuniform	PM-AdapSS-DE
$(1,4)^m_s$CMA-ES	DIRECT	POEMS
$(1,4)_s$CMA-ES	EDA-PSO	PSO with bounds
$(1+1)$CMA-ES	FULLNEWUOA	PSO without bounds
$(1+2)^m_s$CMA-ES	G3PCX	RCGA
ABC	GA	Rosenbrock
ALPS	GLOBAL	VNS
AMALGAM	IPOP-SEP-CMA-ES	iAMALGAM
BAYEDA	LSfminbnd	oPOEMS
BFGS	LSstep	pPOEMS
BIPOP-CMA-ES	MA-LS-CHAIN	

$$\widehat{T}\left(f,\alpha,e_t\right)=\log_{10}\left(\frac{1}{D}\frac{\#\text{FEs}\left(\left(y_b-y_t\right)\geq e_t\right)}{\#\text{succ}}\right) \quad (1)$$

where e_t is the target precision, $\#\text{FEs}\left(\left(y_b-y_t\right)\geq e_t\right)$ is the number of function evaluations over all runs where the best cost, y_b, is greater than the optimal cost, and $\#$succ is the number of successful runs. When some runs are unsuccessful, $\widehat{T}$ depends on the termination criteria of the algorithm. The average expected running time over a subset of problems, F, is defined as:

$$\overline{T}\left(F,\alpha,e_t\right)=\frac{1}{|F|}\sum_{f\in F}\widehat{T}(f,\alpha,e_t) \quad (2)$$

To measure the accuracy of the algorithm or portfolio, we use the success rate (SR), which is the percentage of solved problems by the algorithm. To summarize the efficiency and accuracy results of the algorithm, we define a normalized performance score, ρ, as follows:

$$\rho\left(F,\alpha,e_t\right)=\frac{\overline{T}\left(F,\alpha_T,e_t\right)}{SR\left(F,\alpha_T,e_t\right)}\cdot\frac{SR\left(F,\alpha,e_t\right)}{\overline{T}\left(F,\alpha,e_t\right)} \quad (3)$$

where α_T is a baseline algorithm for which $\rho=1$. We define e_t to be equal to 10^{-8}.

D. Building the portfolio

To build a portfolio with ICARUS, we carry out a parameter sweep for K in the range $[1,|\mathcal{A}|]$, where $|\mathcal{A}|=47$ is the number of algorithms in Table I. For each value of K, we calculate the size of A and $\overline{T}$ of the best algorithm in the portfolio. Given that the results of ICARUS depend on the performance data; the value of K selected through this method would be valid only for this experiment.

Two validation approaches are followed: the hold-one-instance-out (HOIO) and hold-one-problem-out (HOPO) proposed in [23]. For HOIO, the data from one instance of each problem at all dimensions is removed from the training set.

TABLE II
PERCENTAGE OF PROBLEMS FROM WHICH EACH ALGORITHM IS THE BEST DEPENDING OF THE PORTFOLIO. THE NUMBERS IN PARENTHESES REPRESENT THE PERCENTAGE OF PROBLEMS FOR WHICH AN ALGORITHM WAS SELECTED BY ICARUS DURING HOPO.

	ICARUS/HOIO	ICARUS/HOPO		BBLL
$(1+2)^m_s$CMA-ES		12.5%	(87.5%)	
BFGS	27.1%			24.0%
BIPOP-CMA-ES	63.5%	43.8%	(100.0%)	61.5%
Lsfminbnd				6.3%
Lsstep	9.4%	10.4%	(100.0%)	8.3%
Nelder-Doerr		33.3%	(100.0%)	

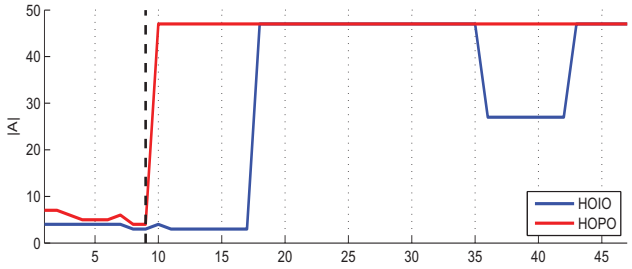
For HOPO, all the instances for one problem at all dimensions are removed from the training set. HOIO measures the performance for instances of problems previously observed; whereas HOPO measures the performance for instances of unobserved problems. The results from the test instances are compiled to obtain a final result. Since F changes for each validation approach, an independent parameter sweep is carried out for HOIO and HOPO resulting in two portfolios. We compare our results with a portfolio composed of the BFGS, BIPOP-CMA-ES, LSfminbnd, and LSstep algorithms (BBLL) [23]. In contrast to ICARUS, this portfolio was manually designed to minimize $\overline{T}$ over all the benchmark functions within a target of 10^{-3}; hence, it allows us to compare manual and automatic portfolio construction.

IV. RESULTS

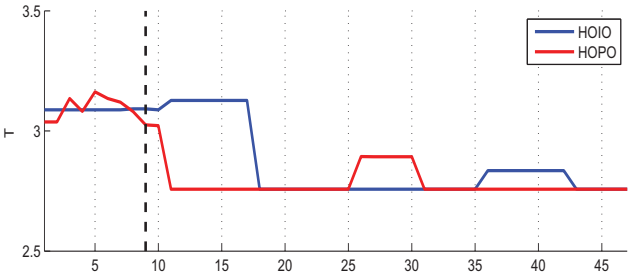
In this section, we present results of the validation process described in Section III. Significantly, ICARUS generates an algorithm portfolio, A, with performance superior to the BBLL portfolio [23].

Figure 1 shows the portfolio size, $|A|$, and the performance of its best algorithm, $\overline{T}$, against the maximum number of valid preferences, K, for both HOIO and HOPO validation. The figure shows that for $K\geq 18$ for HOIO and $K\geq 10$ for HOPO, all the algorithms in $\mathcal{A}$ are selected, resulting in the lowest possible $\overline{T}$. The smallest size of A is three for HOIO and four for HOPO, which is obtained with $K=9$. The highest $\overline{T}$ for HOIO is obtained with $K\in[11,17]$, resulting in $|A|=3$. On the other hand, the highest $\overline{T}$ for HOPO is obtained with $K=5$, resulting in $|A|=5$. To obtain a small A for both HOIO and HOPO, values of K less than ten are feasible. Hence, we set $K=9$ as it gives the best performance with HOPO. The difference in $\overline{T}$ with the best portfolio for HOIO, obtained with $K=1$, is 1.009 function evaluations per dimension (f_{evals}/D). Table II shows the resulting portfolios with the percentage of problems for which each algorithm is the best. The numbers in parentheses represent the percentage of problems for which an algorithm is on ICARUS/HOPO portfolio. All the algorithms selected by ICARUS are part of the BBLL set, except for $(1+2)^m_s$CMA-ES.

The performance of the best algorithm for a given problem is illustrated in Figure 2 for the ICARUS/HOPO and BBLL portfolios. We observe that BIPOP-CMA-ES is the dominant algorithm for $\{10,20\}$ dimensions in both portfolios. The



(a) Portfolio size, $|A|$



(b) Average expected running time, $\overline{T}$

Fig. 1. Parameter sweep for the maximum number of valid preferences, K, of ICARUS. The figure shows that for $K = 9$, the oracle has the best $\overline{T}$ and the smallest $|A|$ for HOPO.

TABLE III
PERFORMANCE OF EACH ALGORITHM AND PORTFOLIO, IN TERMS OF $\overline{T}$, 95% CONFIDENCE INTERVAL OF $\widehat{T}$, SR AND ρ.

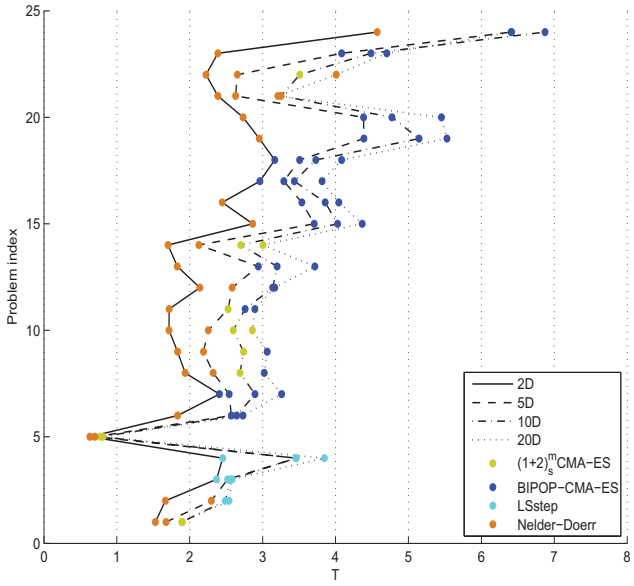
		$\overline{T}$	95% CI $\widehat{T}$	SR	ρ
$(1+2)^m_s$CMA-ES		2.852	$[0.781, 5.159]$	66.7%	0.829
BFGS		2.633	$[0.349, 4.648]$	43.8%	0.589
BIPOP-CMA-ES		3.397	$[1.107, 6.411]$	95.8%	1.000
Lsfminbnd		3.092	$[1.105, 5.172]$	27.1%	0.310
Lsstep		3.306	$[2.085, 5.169]$	30.2%	0.324
Nelder-Doerr		2.765	$[0.694, 5.455]$	59.4%	0.761
BBLL		3.040	$[0.410, 6.410]$	100.0%	1.146
ICARUS	HOIO	3.092	$[0.410, 6.410]$	100.0%	**1.172**
	HOPO	3.025	$[0.770, 6.410]$	100.0%	1.166

Nelder-Doerr algorithm replaces BFGS in the ICARUS/HOPO portfolio, resulting in deterioration of $\widehat{T}$ for $\{f_1, f_5\}$. There are specialized algorithms for particular problems regardless of the number of dimensions, e.g., LSstep is the best for $\{f_3, f_4\}$.

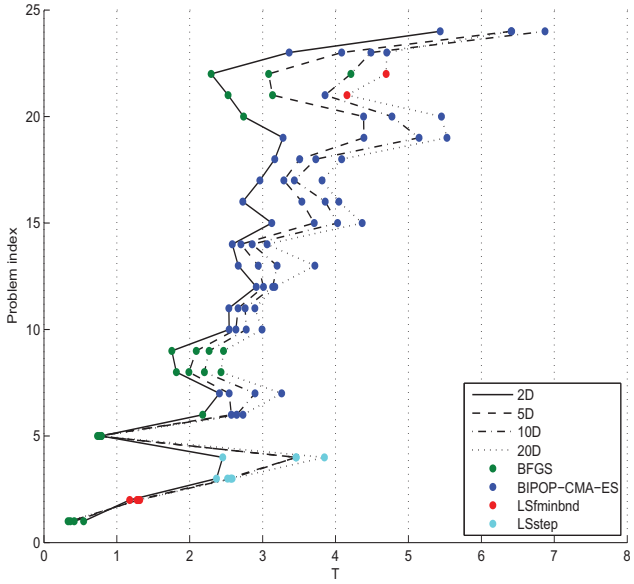
Table III shows the performance of the individual algorithms and portfolios during HOIO and HOPO. The performance is measured as $\overline{T}$, 95% confidence interval (CI) of $\widehat{T}$, SR and ρ. In bold-face are the best values of each performance measure. The table shows the ICARUS portfolio having the best overall performance with $\rho = 1.172$ during HOPO, outperforming the BBLL portfolio.

V. CONCLUSION

In this paper, we have described the design, implementation and evaluation of novel voting-based model used to automatically construct algorithm portfolios for BBO problems. The rationale behind our approach is that all the elements in



(a) ICARUS/HOPO portfolio



(b) BBLL portfolio

Fig. 2. Performance of the best algorithm in terms of $\widehat{T}$ for each problem. The lines represent the dimension of the problem. The Nelder-Doerr algorithm replaces BFGS in the ICARUS/HOPO portfolio in most of the problems.

the uncovered set are Pareto optimal; hence, the set would represent the most reasonable outcome in the portfolio design process [20]. ICARUS uses the $\widehat{T}$ as criteria to select the algorithms at a given target value, e_t, which we set at 10^{-8}. The results presented in this paper are encouraging. However, ICARUS is still at the proof-of-concept stage and further analysis may be necessary.

We have identified two main limitations of ICARUS. The first limitation may be attributed to the selection criteria. ICARUS disregards the possibility that the difference in $\widehat{T}$

between two algorithms is not statistically significant. Therefore, two algorithms may tie on a problem even though their $\widehat{T}$ is numerically different. These ties might be identified using the bootstrap distribution of $\widehat{T}$ and a test of statistical significance, such as the Wilcoxon rank-sum test. To add this information to ICARUS, the matrix of votes, $\mathbf{C}$, should be modified by adding votes to the tied algorithms. The second limitation of ICARUS is due to the uniform distance between preferences. One algorithm may be preferred over another and have a difference of several orders of magnitude in their performance. How to include this information in the rankings remains an open question. The rankings could be improved if these limitations are addressed.

One further point worthy of discussion is the fact that we have not examined the computational cost required to construct the knowledge base of algorithm performance, which is fundamental for running ICARUS. This cost is reduced by using shorter algorithm runs with a less stringent target value, as proposed in [12], [24], [25]. Furthermore, by setting $K = 9$, we obtained the smallest algorithm subset for this particular experiment. However, it excludes algorithms that may well prove to be useful when confronted with unobserved problems (this in turn may improve SR). Further investigation of the trade-off between K and SR is left for further research.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

REFERENCES

[1] M. Lozano, D. Molina, and F. Herrera, "Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems," *Soft Comput.*, vol. 15, pp. 2085–2087, 2011.

[2] P. Hough and P. Williams, "Modern machine learning for automatic optimization algorithm selection," in *Proceedings of the INFORMS Artificial Intelligence and Data Mining Workshop*, 2006, pp. 1–6.

[3] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, April 1997.

[4] W. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 561–578, October 2007.

[5] A. Auger and O. Teytaud, "Continuous lunches are free plus the design of optimal optimization algorithms," *Algorithmica*, vol. 57, no. 1, pp. 121–146, 2010.

[6] K. Sörensen, "Metaheuristics— the metaphor exposed," *Int. T. Oper. Res.*, vol. 22, no. 1, pp. 3–18, January 2015.

[7] D. Weyland, "A rigorous analysis of the harmony search algorithm: How the research community can be misled by a "novel" methodology," *Int. J. Appl. Metaheuristic Comput.*, vol. 1, no. 2, pp. 50–60, 2010.

[8] M. Črepinšek, S. Liu, and L. Mernik, "A note on teaching-learning-based optimization algorithm," *Inform. Sciences*, vol. 212, pp. 79 – 93, 2012.

[9] A. De Corte and K. Sörensen, "Optimisation of gravity-fed water distribution network design: A critical review," *Eur. J. Oper. Res.*, vol. 228, no. 1, pp. 1–10, 2013.

[10] A. Piotrowski, J. Napiorkowski, and P. Rowinski, "How novel is the "novel" black hole optimization approach?" *Inform. Sciences*, vol. 267, pp. 191–200, 2014.

[11] P. Oliveto and X. Yao, "Runtime analysis of evolutionary algorithms for discrete optimization," in *Theory of randomized search heuristics*. World Scientific, 2011, pp. 21–52.

[12] K. Tang, F. Peng, G. Chen, and X. Yao, "Population-based algorithm portfolios with automated constituent algorithms selection," *Inform. Sciences*, vol. 279, pp. 94–104, 2014.

[13] C. Blum, J. Puchinger, G. Raidl, and A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *Appl. Soft Comput.*, vol. 11, no. 6, pp. 4135 – 4151, 2011.

[14] M. Gagliolo and J. Schmidhuber, "Learning dynamic algorithm portfolios," *Ann. Math. Artif. Intel.*, vol. 47, pp. 295–328, 2006.

[15] C. Gomes and B. Selman, "Algorithm portfolios," *Artif. Intell.*, vol. 126, no. 1–2, pp. 43–62, 2001.

[16] S. Y. Yuen and X. Zhang, "On composing an algorithm portfolio," *Memetic Computing*, pp. 1–12, 2015.

[17] S. Y. Yuen, C. K. Chow, X. Zhang, and Y. Lou, "Which algorithm should i choose: An evolutionary algorithm portfolio approach," *Applied Soft Computing*, vol. 40, pp. 654–673, 2016.

[18] E. Penn, "Alternate definitions of the uncovered set and their implications," *Soc. Choice Welfare*, vol. 27, pp. 83–87, 2006.

[19] K. A. Shepsle and B. R. Weingast, "Uncovered sets and sophisticated voting outcomes with implications for agenda institutions," *American Journal of Political Science*, pp. 49–74, 1984.

[20] S. Feld, B. Grofman, R. Hartly, M. Kilgour, and N. Miller, "The uncovered set in spatial voting games," *Theory and Decision*, vol. 23, no. 2, pp. 129–155, 1987. [Online]. Available: http://dx.doi.org/10.1007/BF00126302

[21] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking BBOB-2010: Experimental setup," INRIA, Tech. Rep. RR-7215, December 2014. [Online]. Available: http://coco.lri.fr/downloads/download15.02/bbobdocexperiment.pdf

[22] T. Bartz-Beielstein, *Experimental Research in Evolutionary Computation*, ser. Nat. Comput. Ser. Springer, 2006.

[23] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß, "Algorithm selection based on exploratory landscape analysis and cost-sensitive learning," in *GECCO '12*. New York, NY, USA: ACM, 2012, pp. 313–320.

[24] T. Carchrae and J. Beck, "Applying machine learning to low-knowledge control of optimization algorihms," *Comput. Intell.*, vol. 21, no. 4, pp. 372–387, 2005.

[25] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "F-Race and iterated F-Race: An overview," in *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuß, Eds. Springer, 2010, pp. 311–336.