Fitness Landscape Based Features for Exploiting Black-Box Optimization Problem Structure

Tinus Abell, Yuri Malitsky and Kevin Tierney

IT University Technical Report Series **TR-2012-163**

ISSN 1600–6100 **December 2012**

ISSN 1600–6100

ISBN 978-87-7949-274-5

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: **+45 72 18 50 00**
Telefax: **+45 72 18 50 01**
Web **`www.itu.dk`**

Fitness Landscape Based Features for Exploiting Black-Box Optimization Problem Structure

Tinus Abell, Yuri Malitsky and Kevin Tierney

Abstract

We present a robust set of features that analyze the fitness landscape of black-box optimization (BBO) problems. We show that these features are effective for training a portfolio algorithm using Instance Specific Algorithm Configuration (ISAC). BBO problems arise in numerous applications, especially in scientific and engineering contexts. BBO problems are characterized by computationally intensive objective functions, which severely limit the number of evaluations that can be performed. We introduce a collection of problem independent features to categorize BBO problems and show how ISAC can be used to select the best minimization search strategy. We test our methodology on data from the GECCO Workshop on Black-box Optimization Benchmarking 2012, which contains 21 state-of-the-art BBO solvers run on 24 well-established BBO functions, and show that ISAC is able to exploit our general, problem independent features and outperform any single solver.

1 Introduction

This paper tackles the challenge of crafting a set of features that can capture the structure of black-box optimization (BBO) problem fitness landscapes for use in portfolio algorithms. BBO problems are defined by:

$$\begin{array}{lll} \min & f(x_1,\ldots,x_n) & \\ \text{s.t.} & l_i \leq x_i \leq u_i & 1 \leq i \leq n \\ & x_i \in \mathbb{R} & 1 \leq i \leq n \end{array}$$

BBO problems are found throughout the scientific and engineering fields, but are difficult to solve due to their expensive objective functions. These problems often arise when an objective is too complicated to be defined by a simple mathematical expression, such as objectives computed by simulations or neural networks, but can also involve difficult to compute expressions that cannot be solved using standard mathematical techniques. Although BBO algorithms do not guarantee discovery of the optimal solution, they are nonetheless an effective tool for finding approximate solutions. Different BBO algorithms can show a wide range of performance across a set of problems. Thus, deciding which solver to apply to a particular problem is a difficult task, especially for practitioners who may not be experts in optimization. Portfolio algorithms, such as the ISAC method, provide a way to automatically choose a solver for a particular BBO instance using offline learning. However, such methods require a set of features that consolidate the relevant attributes of a BBO instance into a vector that can then be used for learning.

Features that are able to capture the structure of BBO problems are essential to portfolio approaches like ISAC, however generating features for BBO problems is particularly challenging because the problem structure, other than the number of dimensions in the problem and the accuracy desired of the solver, is hidden within the black-box. Thus, the only way to compute features is to sample the black-box, which is an expensive way of gathering features. This contrasts with most non-black-box problems, e.g. SAT or the set covering problem, where many features can be quickly inferred from the problem definition itself.

Given a training set of instances and a collection of solvers, ISAC tries to find an assignment of solver to instance that maximizes the quality of a performance metric. Unlike the problems ISAC has previously been used on, such as SAT [13], set covering/partitioning [16], mixed-integer programming [13], and constraint programming [14], BBO problems have an inherently instable feature set, since the features must be computed through sampling the fitness landscape. Additionally, the features are significantly more expensive to compute than in other problems.

In this paper, we propose a novel set of features that are fast to compute and are descriptive enough of the instance structure to allow ISAC to accurately cluster and tune for the benchmark. These features are based on well-known fitness landscape measures and are learned through sampling the black box. They allow for the analysis and classification of BBO problems so that anybody can take advantage of the recent advances in the ISAC framework in order to more efficiently solve their BBO problems.

1.1 Related work

There has been extensive research studying the structure of BBO problems, and copious measures have been proposed for determining the hardness of local search problems by sampling their fitness landscape [11], such as the search space diameter, optimal solution density/distribution [6], local optima networks [23], fitness-distance correlation (FDC) [12], the correlation length [25, 28], epistasis measures [21], information analysis [26], modality and neutrality measures [24], and fitness-distance analysis [20]. Difficulty measures for BBO problems in particular were studied by [10], who concluded that in the worst case building predictive difficulty measures for BBO problems is not possible to do in polynomial time[1]. Most recently, Watson introduced several cost models for combinatorial landscapes in order to try to understand why certain algorithms perform well on certain landscapes [27].

In [18], the authors identify six "low-level feature classes" to classify BBO problems into groups. In [4], algorithm selection for BBO problems is considered with a focus on minimizing the cost of incorrect algorithm selections, unlike our approach, which minimizes a score based on the penalized expected runtime. In addition, our approach evaluates our features on the state-of-the-art ISAC method. Our approach also differs from online methods [5] and reactive techniques [3] that attempt to guide algorithms based on information from previously explored states because ISAC performs all of its work offline. Our approach differs from online methods [5] and reactive techniques [3] that attempt to guide algorithms based on information from previously explored states because ISAC performs all of its work offline. Additionally, neither of these approaches uses a feature set based on the landscape structure.

2 ISAC

ISAC is a recently proposed approach for selecting solvers from a portfolio in order to more efficiently solve a set of instances. Unlike similar approaches, such as Hydra [29] and ArgoSmart [22], ISAC does not use regression-based analysis. ISAC computes a representative feature vector in order to identify clusters of similar instances. The data is therefore clustered into non-overlapping groups and a single solver is chosen for each group based on some performance characteristic. Given a new instance, its features are computed and it is assigned to the nearest cluster. The instance is then solved with the solver assigned to that cluster.

ISAC works as follows (see Algorithm 1). In the learning phase, ISAC is provided with a parameterized solver A, a list of training instances T, their corresponding feature vectors F, and the minimum cluster size κ. In the case of a portfolio algorithm, as in this paper, the metasolver, A, has a single parameter which specifies which solver should be used. First, the gathered features are normalized so that every feature ranges from $[-1, 1]$, and the scaling and translation values for each feature (s, t) is memorized. This normalization helps keep all the features at the same order of magnitude, and thereby keeps the larger values from being given more weight than the lower values. Then, the instances are clustered based on the normalized feature vectors. Clustering is advantageous because it helps prevent overfitting by ensuring that solvers work across a number of instances. This helps reduce the influence of noisy data in the training set.

To avoid specifying the desired number of clusters beforehand, the g-means [9] algorithm is used. Robust parameter sets are obtained by not allowing clusters to contain fewer than a manually chosen threshold, κ, a value which depends on the size of the data set. In the case of this paper, we experimented with several limits on the number of instances per cluster, showing that setting it correctly is important for good performance from ISAC. Beginning with the smallest cluster, the corresponding instances are redistributed to the nearest clusters, where proximity is measured by the Euclidean distance of each instance to the cluster's center. The final result of the clustering is a number of k clusters S_i, and a list of cluster centers C_i. Then, for each cluster of instances S_i, favorable parameters P_i are

[1] We note that our results do not contradict this conclusion, as we are not predicting the hardness of instances.

Algorithm 1 Instance-Specific Algorithm Configuration

ISAC-Learn(A, T, F, κ)
$(\bar{F}, s, t) \leftarrow$ Normalize(F)
$(k, C, S) \leftarrow$ Cluster $(T, \bar{F}, \kappa)$
for all $i = 1, \ldots, k$ **do**
  $P_i \leftarrow Best(A, S_i)$
end for
return (k, P, C, s, t)

ISAC-Run(A, x, k, P, C, d, s, t)
$f \leftarrow$ Features(x)
$\bar{f}_i \leftarrow 2(f_i/s_i) - t_i \; \forall \; i$
$i \leftarrow \min_i(||\bar{f} - C_i||)$
return $A(x, P_i)$

computed. ISAC was first introduced using the GGA parameter tuner [1] on a single paramterized algorithm, however in the case of a portfolio, the tuning step can be considered to consist of selecting the solver that provides the best performance on the cluster.

When running algorithm A on an input instance x, ISAC first computes the features of the input and normalize them using the previously stored scaling and translation values for each feature. Then, the instance is assigned to the nearest cluster. Finally, ISAC runs A on x using the solver for this cluster.

3 BBO Dataset

We evaluate the effectiveness and robustness of our features on a dataset from the GECCO 2012 Workshop on Black-Box Optimization Benchmarking (BBOB) [2]. The dataset contains the number of evaluations required to find a particular objective value within some precision on one of 24 continuous, noise-free, optimization functions from [8] in 6 different dimension settings for 27 solvers. The solvers are all run on the data 15 times, each time with a different target value set as the artificial global optimum. Note that the BBOB documentation refers to each of these target values as an "instance". To avoid confusion with the instances that ISAC uses to train and test on, we will only refer to BBOB targets.

The dataset includes a number of well-known continuous functions, like the Rastrigin, Rosenbrock and Schwefel functions, and thus represents a high quality benchmark for the effectiveness of our features on BBO problems. The functions range from unimodal to multi-modal, with some having extremely rugged landscapes. Although all of our BBO problems are defined by mathematical expressions, they are treated as a black-box and our features do not use any information about the expression.

Each instance in our dataset is defined by a triple $\langle f, d, p \rangle$, where $f \in \{1, \ldots, 24\}$ is the function, $d \in \{2, 3, 5, 10, 20, 40\}$ is the number of dimensions, and $p \in \{1 \times 10^{-i} \text{ for } i = 0 \ldots 8\}$ is the precision demanded of the solver. After removing 7 instances from the dataset for which no solver was able to find a solution, the dataset consists of 1289 instances.

Using this data, we compute the *expected running time* (ERT) [7] for each solver on each function and dimension at a specified precision. The ERT refers to the expected number of evaluations necessary to reach a target function value, including restarts. Each ERT is based on all 15 BBOB targets of a function on a particular function-dimension pair.

3.1 Solver portfolio

ISAC requires a portfolio consisting either of highly parameterized solvers, or a diversity of solvers. Our dataset is fixed in the sense that we do not have access to the underlying solvers, i.e. the dataset consists of the amount of evaluations required for each solver on each instance. We are therefore unable to exploit the parameterization

of solvers. However, the diversity of solvers in the BBOB dataset is more than satisfactory. Although there are 27 solvers in total in the dataset, we remove solvers from the dataset that do not have solution data for all instances. This leaves 21 solvers, more than enough for ISAC. The portfolio of solvers consists of a diverse set of continuous optimizers, including 10 covariance matrix adaptation (CMA) variants, 8 differential evolution (DE) variants, an ant colony optimization (ACO) algorithm, a genetic algorithm (GA), and a particle swarm optimization (PSO) algorithm[2]. Very few of these solvers are "pure" implementations, in that they include either advanced heuristics or are hybridized with other algorithms, in particular BFGS [15]. No single solver dominates the dataset; in other words, no solver has such high performance that it outperforms all of the other solvers on all instances, meaning the BBOB data is ideal for use by ISAC.

4 Features

There are two key difficulties in computing features for BBO problems, the fact that there is scarce information about a problem instance given in the instance definition, and that evaluating the objective of a BBO problem is expensive. BBO problems offer no hints about their underlying structure up front (e.g. in a problem definition file), other than the number of dimensions and the accuracy desired of the solver. For example, features for SAT problems include a deep analysis of the relationship between variables and clauses using graphs, among other features [30]. In SAT, accessing this information is cheap, since it is a part of the specification of a SAT problem. In the absence of any structure in the problem definition, we have no choice but to sample the fitness landscape to try to understand the landscape. However, sampling the landscape is expensive, and therefore the amount of sampling that can be performed is severely limited in comparison to the sampling that is performed in other domains. In fact, performing more than about 600 objective evaluations results in so much time being spent on feature computation, that using a portfolio approach is no longer worthwhile. It is therefore critical that our features extract as much information about the search landscape as possible, while curtailing the number of evaluations to the maximum extent. With these constraints in mind, we introduce a set of 10 features describing BBO problems that are based on well-studied aspects of search landscapes in the literature [27]. Our features are divided into three categories: problem definition features, hill climbing features, and random point features. Our features are summarized in Table 1 and described below.

Problem definition features These features contain all the data that we can possibly extract from the problem itself: the desired accuracy of the continuous variables (Feature 1), and the number of dimensions that the problem has (Feature 2). The number of dimensions and the desired accuracy together describe the size of the problem.

Hill climbing features These features perform a number of hill climbs that are initiated from random points and continued until a local optimum or a fixed number of evaluations is reached. We then calculate the average and standard deviation of the distance between optima (Features 3 and 4), which describes the density of optima in the landscape. Using the best optimum found, we then compute the average and standard deviation of the distance between the optima and the best optimum (Features 5 and 6). If multiple optima qualify as the best, we use the nearest to each non-best optimum for these features. Feature 7 describes what percentage of the optima are equal to the best optimum, which indicates whether the optima are all clustered near each other, or if they are spread out through the landscape. Some BBO problems have large convex areas, in which multiple hill climbs will lead to the same position. Thus, high values of feature 7 signal relatively convex landscapes.

Random point features Features 8 and 9 contain the average and standard deviation of the distance of each random point to the nearest optimum, which describes the distribution of local optima around the landscape. Feature 10 computes the fitness-distance correlation, a measure of how effectively the fitness value at a particular point can guide the search to a global optimum [12, 19]. In feature 10, we compute an approximation to the FDC[3] using a set of

[2]Full details about the algorithms are available in [2]

[3]Note that computing the exact FDC requires complete knowledge of the search space, including the global minimum. With such knowledge running ISAC would be pointless.

Problem definition features	
1.	Solver accuracy
2.	Number of dimensions
Hill climbing features	
3-4.	Average distance between optima (average, std. dev.)
5-6.	Distance between best optima and other optima (average, std. dev.)
7.	Percent of optima that are the best optimum
Random point features	
8-9.	Distance to local optimum (average, std. dev.)
10.	Fitness-distance correlation (FDC)

Table 1: BBO problem features.

randomly sampled points, where we denote s^* as the optimum found by hill climbing with the minimal fitness value. Thus

$$FDC = \frac{Cov(g,d)}{\sigma(g) \cdot \sigma(d)}$$

where g is a vector containing the fitness evaluations of the random points, d is a vector containing the distance of each random point to s^*, $Cov(g,d)$ is the covariance of g and d, and $\sigma(g)$ ($\sigma(d)$) is the standard deviation of the random point objectives (distances).

4.1 Further potential features

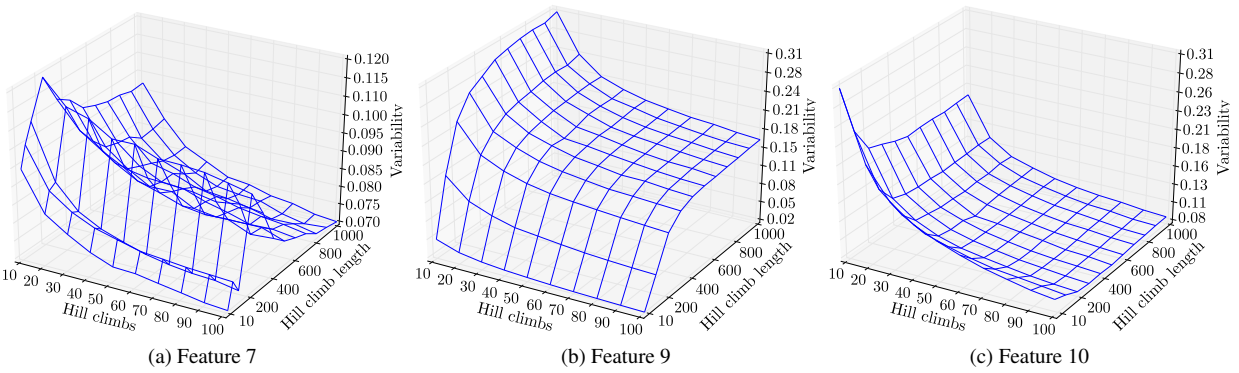
There are a number of search landscape properties that have been investigated in the literature that would be candidates for use as BBO features if not for their excessive number of objective evaluations. The first possible features are the search position types as described in [11], which require at least two evaluations for each dimension for each random point analyzed. Second, the *correlation length* of landscape [25, 11, 27], which measures a landscape's ruggedness, could be used to assess the local optima density in the landscape. Finally, many aspects of landscapes have been investigated that require complete or near-complete knowledge of the local optima, such as local optima networks [23], which creates a compact structure representing the local optima in a landscape, or determining exact local optima density in regions of the landscape. Although such measures can provide valuable information about the search space, the goal of computing features is to quickly determine what solver can best solve a problem instance, not to solve the instance with the feature computations. Note that these measures stand in contrast to the FDC, which is an approximation, and does not require a full view of the landscape.

4.2 Feature computation

In order to conserve objective evaluations during the computation of our features for our robustness experiments, as well as in our numerical results, our set of random points only includes the first point on each hill-climb, as these points are chosen uniformly at random and must be evaluated for each hill climb anyway. While there is a risk that this will bias some of our features, in particular the distances to the nearest local optimum in features 8 and 9, the gains in evaluations are worth such a bias. The alternative to this would be to select random points uniformly at random, separate from the hill climbs. We further consider any local optimum with an objective within 5% of the best optimum to be equivalent to the best optimum.

4.3 Feature robustness

Since our features are inherently stochastic, whether the features are robust or not becomes an important question. If the feature computation greatly varies each time it is run, it is difficult to learn an effective model to predict which solver will solve which instance most effectively. Within the ISAC approach, features that vary greatly could move instances between clusters and negatively effect performance.



(a) Feature 7 (b) Feature 9 (c) Feature 10

Figure 1: Measurements of the variability in our features with varying parameters.

Figure 1 shows the *variability* of several features on the functions in our dataset. For each graph, the variability is computed as follows. We first compute the features for each combination of number of hill climbs and maximum hill climb length. For a given feature, we then compute the standard deviation of the feature on each instance across all 15 BBOB targets. Finally, we take the average of the standard deviations from each instance, giving us a measure of how much each feature varies over the dataset.

Three basic pictures emerge from this experiment. Feature 7 (Figure 1a) has an area of increased noisiness when the maximum hill climb length is set to 300. Note that this feature measures the percent of optima that are the same as the best optimum (within 5% of the objective value). When the hill climb length is short (less than 300), we suspect that the hill climb does not go deep enough to cause much variation in the number of optima that match the best optimum. When the hill climb is fairly long (greater than 300), the hill climb is able to find or come close to the bottom of a basin of attraction, leading to a stable feature. Around length 300, however, the hill climb has not quite made it far enough to result in a stable feature. Features 3, 4, 5, 6 and 8 resemble feature 9 (Figure 1b). The variability for these features is characterized by highly stable features for short hill climb lengths, followed by a decrease in stability, and then a plateau above length 400. We attribute this behavior to the fact that hill climbs under length 400 simply are not long enough to yield much information, resulting in features that do not vary much, but are not particularly useful. Feature 10 (Figure 1c), which relies on both random points and hill climbs, has high variability with a low hill climb length and becomes more stable as the hill climb length (and number of hill climbs) increases. This is because the FDC measurement in feature 10 uses the best objective value found for its computations. When the hill climbs are short, the best objective value bounces around the landscape. This influences the distances to this value, causing them to change each time the features are computed, resulting in an unstable feature. Feature 10 stabilizes after about 30 hill climbs or length 200 hill climbs. An important, and not surprising, trend present in all of the features is that low number of hill climbs lead to unstable features, whereas more hill climbs lead to stable features. Furthermore, it is clear that the number of hill climbs has a larger effect on the stability of features than the length of the hill climb. We forego an in depth feature filtering using the methods from [14], despite its success in filtering constraint programming features for ISAC, due to the low number of features we are using, and due to the relative stability of all of our features across the dataset.

5 Numerical Results

In this section we describe the results of using our features, in full and in various combinations, to train a portfolio solver using the ISAC method on the BBOB 2012 dataset.

5.1 Scoring and Penalized ERT

We measure the performance of each solver using a penalized score that takes into account the relative performance of each solver on an instance. We do not directly use the ERT value because the amount of evaluations can vary greatly

between instances, and too much focus would be placed on instances where a large number of evaluations is required. In the case of low dimensional instances, which do not generally require many evaluations, the choice of solver would essentially be irrelevant and the portfolio would ignore them. While there are settings where this might be desirable, such as for a practitioner who has thousands of instances to solve, our goal is to create a portfolio that can efficiently solve *any* instance given to it. Thus, if a practitioner were to solve several instances with a randomly distributed size, our portfolio would choose better solvers for all of those instances than if the practitioner had used a single solver. The penalized score of solver s on an instance i is given by:

$$score(s,i) = \frac{PERT(s,i) - best(s,i)}{worst(s,i) - best(s,i)}$$

where $PERT(s,i)$ is the penalized ERT defined by

$$PERT(s,i) = \begin{cases} ERT(s,i) & \text{if } ERT(s,i) < \infty \\ worst(s,i) \cdot 10 & \text{otherwise,} \end{cases}$$

$best(s,i)$ refers to the lowest ERT score on instance i, and $worst(s,i)$ refers to the highest non-infinity ERT score on the instance. The penalized ERT therefore returns ten times the worst ERT on an instance for solvers that were unable to find the global optimum. We are forced to use a penalized measure because if a solver cannot solve a particular instance, it becomes impossible to calculate its performance over the entire dataset. Using a penalization of factor of 10 is standard for penalized average runtime (PAR) scores in the literature [29]. The goal of such a high penalization factor is to discourage ISAC from choosing solvers that cannot solve certain instances.

5.2 ISAC Results

Table 2 shows the results of training and testing ISAC on the BBOB 2012 dataset. For each entry in the table, we run a 10-fold cross validation using features from each of the 15 BBOB target values. The scores of each of the cross-validation folds are accumulated for each instance, and the entries in the table are the average and standard deviation across all instances in the dataset. We compare our results against the best single solver (BSS) on the dataset, which is simply the best performing solver across all instances. In our experiments, the solver MVDE [17], a differential evolution algorithm that applies multiple mutations to the population and merges them using a tournament, provided the best score across all instances. We test several different subsets of our features. First, we train ISAC using only feature 1 (F1), only feature 2 (F2), and only features 1 and 2 (F(1,2)) in order to ascertain whether we can simply use easy to acquire information about BBO problems, or if a deeper landscape analysis is required. We then train using all features (All), and only landscape features (LSF), i.e., features 3 through 10. All* and LFS* include the evaluations necessary to compute the features, whereas all other entries do not include the feature computation in the results. We used several different settings of the number of hill climbs and maximum hill climb length: 10 hill climbs of maximum length 10, 50 hill climbs of maximum length 20, and 200 hill climbs of maximum length 400. We choose these parameterizations of the feature computation to show the performance of the features under varying numbers of evaluations. In particular, we picked 200 hill climbs of length 400 based on our feature robustness experiments, which emphasize the importance of the number of hill climbs, as well as having over a hill climb length of over 400 or more. The closer a score is to 0, which is the score of the virtual best solver, the better the performance of an approach.

We report results for F1, F2 and F(1,2) in order to show that our easy to compute BBO features alone are only able to give ISAC some information about the dataset, and that a landscape analysis is justified. Note that F1, F2 and F(1,2) do not rely on the number of hill climbs or the hill climb length. It is immediately clear that F1 and F(1,2) are not able to create an effective predictive model. Although the training performance of F1 and F(1,2) is on average equal to or better than BSS, this does not translate into good test performance, with the performance score of F1 and F(1,2) being several orders of magnitude higher than BSS. On the other hand, F2, which is the number of problem dimensions, outperforms BSS both for cluster size 50 and 100. In fact, F2 performs as equally well as All and LSF for cluster 100 with 10 hill climbs of length 10 and for 50 hill climbs of length 20. In addition, F2 significantly outperforms All on cluster size 50, where it is clear that it overfits the training data. This is a clear indication that 10 hill climbs of length 10, or 50 hill climbs of length 20, do not provide enough information to train ISAC to be competitive with simply using the number of dimensions of a problem.

κ		10/10				50/20				200/400			
		Test		Train		Test		Train		Test		Train	
		⊘	σ	⊘	σ	⊘	σ	⊘	σ	⊘	σ	⊘	σ
50	BSS	2.23	5.29	2.23	5.29	2.23	5.29	2.23	5.29	2.23	5.29	2.23	5.29
	F1	2474.63	30053.87	2.04	5.08	2474.66	30053.87	2.04	5.08	2474.64	30053.87	2.04	5.08
	F2	1.24	4.02	1.24	4.02	1.24	4.02	1.24	4.02	1.24	4.02	1.24	4.02
	F(1,2)	189.11	6743.81	1.27	4.07	189.10	6743.81	1.27	4.07	189.10	6743.81	1.26	4.07
	All	51.32	1801.27	1.21	3.96	96.15	3105.76	0.79	2.94	13.41	452.79	0.82	3.30
	All*	51.42	1801.33	1.32	4.05	97.15	3110.46	1.82	9.90	95.25	1161.92	83.12	760.37
	LSF	1.25	4.01	1.24	4.00	88.18	3137.52	0.82	3.03	**0.53**	2.73	0.55	2.75
	LSF*	1.35	4.09	1.34	4.08	89.18	3142.23	1.85	9.93	99.44	1323.68	82.86	760.40
100	BSS	2.23	5.29	2.23	5.29	2.23	5.29	2.23	5.29	2.23	5.29	2.23	5.29
	F1	2474.63	30053.87	2.04	5.08	2474.66	30053.87	2.04	5.08	2474.64	30053.87	2.04	5.08
	F2	1.24	4.02	1.24	4.02	1.24	4.02	1.24	4.02	1.24	4.02	1.24	4.02
	F(1,2)	189.11	6743.81	1.27	4.07	189.11	6743.81	1.27	4.07	189.10	6743.81	1.27	4.07
	All	1.25	4.02	1.24	4.00	1.25	4.03	1.23	4.00	1.16	3.86	1.12	3.80
	All*	1.35	4.10	1.34	4.08	2.28	10.21	2.26	10.20	83.46	760.60	83.43	760.34
	LSF	1.25	4.02	1.24	4.01	1.22	3.99	1.19	3.93	1.20	3.85	1.15	4.00
	LSF*	1.35	4.10	1.34	4.09	2.25	10.19	2.22	10.17	97.31	1223.98	83.45	760.34

Table 2: The average and standard deviation of the scores across all instances for various minimum cluster sizes, numbers of hill climbs and hill climb lengths for the best single solver and ISAC using various features.

The fact that LSF* is able to match the performance of F2 on 10 hill climbs of length 10 for both cluster size 50 and 100 an important accomplishment. With so little information learned about the landscape, the fact that ISAC can learn such an effective model indicates that our features are indeed effective.

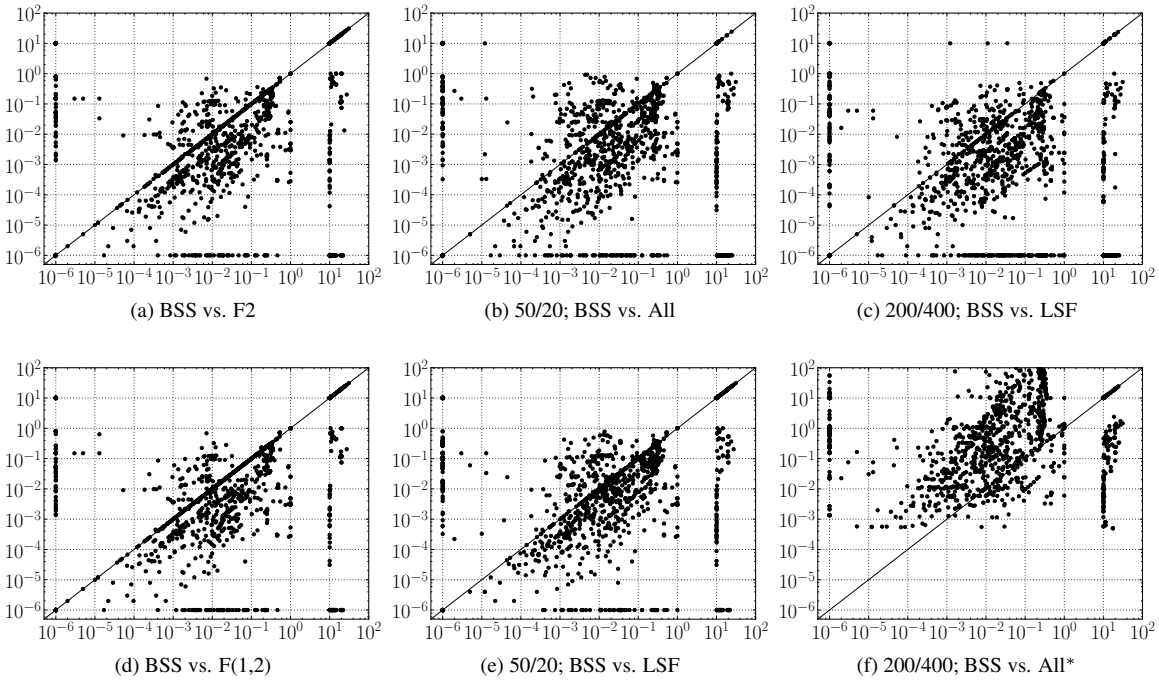
Once we move up to 200 hill climbs of length 400, LSF significantly outperforms F2, and even outperforms All, which suffers from overfitting. In fact, LSF is able to cut the total score to under a fourth of BSS's score, and to one half of F2's score, indicating that the fitness landscape can indeed be used for a portfolio. In addition, LSF has a lower standard deviation than BSS. LSF's score on the training set of 0.53 and 0.55 on the test set are surprisingly close to the virtual best solver, which has a score of zero, indicating that ISAC is able to exploit the landscape features to nearly always choose the best or second best solver for each instance. On the downside, 200 hill climbs of length 400 requires too many evaluations to be used in a competitive portfolio, and All* needs 50 times the evaluations of BSS. However, the 200/400 features are still useful for classifying instances into groups and analyzing the landscape.

Overall, cluster size does have an impact on ISAC performance, and seems to be a factor in the overfitting behavior that occurs several times, such as for All and LSF on cluster size 50 with 50 hill climbs of length 20 and All on 10 hill climbs of length 10. It has been observed that cluster size is important for preventing overfitting with ISAC [13], and our results reaffirm this finding.

Note that we do not provide information from feature gathering to the solvers, since the dataset from BBOB 2012 is fixed. The improvements of LSF over BSS thus represent only a lower bound of what might be accomplished if information gained during feature gathering could be used in the algorithm itself. Since the focus of this paper is primarily on evaluating whether the features work, we save this for future work.

Several entries in Table 2 show poor performance for the features in the average case, mainly due to a couple of BBO targets having rather poor performance. Figure 2 displays the scores of test instances of several portfolios against BSS on BBOB target 1, showing that while the average performance of some features and hill climb/hill climb length combinations is not good, individual BBOB targets do get rather good results. The figure shows the scores of each test instance of BSS against F2 (Figure 2a), All with 50 hill climbs of length 20 (Figure 2b), and LSF with 200 hill climbs of length 400 (Figure 2c), F(1,2) (Figure 2d), LSF with 50 hill climbs of length 20 (Figure 2e), and All* with 200 hill climbs of length 400. Figures 2 a, b and c use a cluster size of 50, and d, e and f a cluster size of 100. All figures are generated from BBOB target 1. Instances are plotted using their BSS score on the x-axis and their score from F2, All or LFS on the y-axis. Thus, BSS is outperformed when a point is above the line $y = x$, and BSS outperforms the other approach when a point is below $y = x$.

In all cases shown for cluster size 50 it is clear that BSS is soundly outperformed, with LSF 200/400 providing the best performance. The good performance of All on 50 hill climbs of length 20 on BBOB target one is in strong



(a) BSS vs. F2 (b) 50/20; BSS vs. All (c) 200/400; BSS vs. LSF

(d) BSS vs. F(1,2) (e) 50/20; BSS vs. LSF (f) 200/400; BSS vs. All*

Figure 2: Comparison of scores against BSS on cluster sizes 50 (a–c) and 100 (d–f).

contrast to the overall result of All on 50 hill climbs of length 20, which shows BSS soundly beating All. This means that the performance can vary greatly across targets, as is seen by the standard deviation of All on 50 hill climbs of length 20, which is over 3000 even though the average score is 96. Thus, our features can actually be predictive and effective, even with an elementary examination of the fitness landscape. Unfortunately, however, these results are not robust, and bad results on the test data are often hidden by strong performance on the training data. We see similar trends on cluster size 100, where F(1,2) is able to outperform BSS, despite poor average performance over all of the BBOB targets. The good performance of LSF with 50 hill climbs of length 20 is clearly displayed, with much of its performance gain coming from choosing solvers that are slightly outperforming the best solver. Finally, the effect of including feature computation evaluations in the overall scoring is displayed for All* with 200 hill climbs of length 400. The added evaluations for computing the features shifts the points up the y-axis, resulting in very few instances outperforming BSS.

Statistical significance We use a two-tailed t-test to determine the statistical significance of our results with the following null hypothesis: The portfolios generated using the features introduced in this paper are not able to outperform BSS on a test set. If the portfolio is able to outperform BSS at a statistically significant level, our features provided useful information to ISAC when it generates a model. We are able to reject the null hypothesis for F2 on both cluster sizes, LSF and LSF* for 10 hill climbs of length 10 and 200 hill climbs of length 400 on clusters 50 and 100 as well as for 50 hill climbs of length 20 on cluster size 100 with a $t < 10^{-4}$, indicating a high level of significance. All and All* also have a $t < 10^{-4}$ on 10 hill climbs of length 10 for cluster size 100, but once the number of evaluations begins to rise in the 50 and 200 hill climb settings, we are no longer able to reject the null hypothesis, except for All on 50 hill climbs of length 20 with a cluster size of 100.

6 Conclusion and Future Work

We introduced a set of features based on accepted and well-studied properties and measures of fitness landscapes to categorize BBO problems for use in algorithm portfolios, like ISAC, that can greatly improve the ability of practitioners to solve BBO problems. We confirmed the robustness of our features through an analysis of the variability of the features across our dataset. Finally, we experimentally validated our features within the ISAC framework, showing that ISAC is able to exploit problem structure learned during feature computation to choose the fastest solver for an unseen instance. The success of the features we introduced clearly indicates that selecting algorithms from a portfolio based on the landscape structure is possible. For future work, features analyzing landscape structure could be incorporated into problems, providing an alternative view of problem structure. Furthermore, problem independent features could be investigated for problems with less difficult objectives where tens or even hundreds of thousands of objective evaluations are feasible. Additionally, we plan to use these features to analyze what types of landscapes fit best to which solvers, which could influence solver development, allowing solvers to more specifically target problems they solve well.

7 Acknowledgements

We would like to thank Dario Pacino for his insightful comments on an early draft of this work. Yuri Malitsky is partially supported by the EU FET grant ICON (project 284715). Kevin Tierney is supported by the Danish Council for Strategic Research as part of the ENERPLAN project.

References

[1] C. Ansotegui, M. Sellmann, and K. Tierney. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. In Ian P. Gent, editor, *CP-09*, volume 5732 of *LNCS*, pages 142–157. Springer, 2009.

[2] A. Auger, N. Hansen, V. Heidrich-Meisner, O. Mersmann, P. Posik, and M. Preuss. GECCO 2012 Workshop on Black-Box Optimization Benchmarking (BBOB). `http://coco.gforge.inria.fr/doku.php?id=bbob-2012`, 2012.

[3] R. Battiti and M. Brunato. Reactive search optimization: learning while optimizing. *Handbook of Metaheuristics*, pages 543–571, 2010.

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

[5] J. Boyan and A.W. Moore. Learning evaluation functions to improve optimization by local search. *The Journal of Machine Learning Research*, 1:77–112, 2001.

[6] C. Brooks and E. Durfee. Using Landscape Theory to Measure Learning Difficulty for Adaptive Agents. In Alonso, E., et. al., editor, *Adaptive Agents and Multi-Agent Systems*, volume 2636 of *LNCS*, pages 561–561. Springer, 2003.

[7] S. Finck, N. Hansen, and R. Ros. Coco documentation, release 11.06, 2012.

[8] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions. Technical report, Technical Report 2009/21, Research Center PPE, 2010.

[9] G. Hamerly and C. Elkan. Learning the k in k-means. *NIPS*, 2003.

[10] J. He, C. Reeves, C. Witt, and X. Yao. A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability. *Evolutionary Computation*, 15:435–443, December 2007.

[11] H.H. Hoos and T. Stützle. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann Publishers Inc., 2004.

[12] T. Jones and S. Forrest. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In *ICGA-95*, pages 184–192, 1995.

[13] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC–Instance-Specific Algorithm Configuration. In *ECAI-10*, volume 215 of *FAIA*, pages 751–756, 2010.

[14] C. Kroer and Y. Malitsky. Feature filtering for instance-specific algorithm configuration. In *23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI11)*, pages 849–855. IEEE, 2011.

[15] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[16] Y. Malitsky and M. Sellmann. Stochastic offline programming. In *21st International Conference on Tools with Artificial Intelligence, 2009. ICTAI'09.*, pages 784–791. IEEE, 2009.

[17] V.V. Melo. Benchmarking the multi-view differential evolution on the noiseless bbob-2012 function testbed. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, GECCO Companion '12, pages 183–188. ACM, 2012.

[18] O. Mersmann, B. Bischl, H. Trautmann, M. Preuß, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *GECCO'11*, pages 829–836. ACM, 2011.

[19] P. Merz and B. Freisleben. Fitness Landscapes and Memetic Algorithm Design. In Corne, D. et. al., editor, *New Ideas in Optimization*, pages 245–260. McGraw–Hill, 1999.

[20] P. Merz and B. Freisleben. Fitness Landscapes, Memetic Algorithms, and Greedy Operators for Graph Bipartitioning. *Evolutionary Computation*, 8:61–91, March 2000.

[21] B. Naudts and L. Kallel. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions On Evolutionary Computation*, 4(1):1 – 15, 2000.

[22] M. Nikolić, F. Marić, and P. Janičić. Instance-Based Selection of Policies for SAT Solvers. In O. Kullmann, editor, *SAT-09*, volume 5584 of *LNCS*, pages 326–340. Springer, 2009.

[23] G. Ochoa, S. Verel, and M. Tomassini. First-improvement vs. best-improvement local optima networks of NK landscapes. *Parallel Problem Solving from Nature – PPSN XI*, pages 104–113, 2011.

[24] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness Landscapes and Evolvability. *Evolutionary Computation*, 10(1):1–34, 2002.

[25] P.F. Stadler and W. Schnabl. The landscape of the traveling salesman problem. *Physics Letters A*, 161(4):337 – 344, 1992.

[26] V.K. Vassilev, T.C. Fogarty, and J.F. Miller. Information Characteristics and the Structure of Landscapes. *Evolutionary Computation*, 8:31–60, March 2000.

[27] J. Watson. An Introduction to Fitness Landscape Analysis and Cost Models for Local Search. In M. Gendreau and J. Potvin, editors, *Handbook of Metaheuristics*, volume 146, pages 599–623. Springer, 2010.

[28] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336, 1990.

[29] L. Xu, H.H. Hoos, and K. Leyton-Brown. Hydra: Automatically Configuring Algorithms for Portfolio-Based Selection. In *AAAI-10*, pages 210–216, 2010.

[30] L. Xu, F. Hutter, H.H. Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of Artificial Intelligence Research*, 32(1):565–606, 2008.