A Meta-learning Prediction Model of Algorithm Performance for Continuous Optimization Problems*

Mario A. Muñoz[1], Michael Kirley[2], and Saman K. Halgamuge[1]

[1] Department of Mechanical Engineering
[2] Department of Computing and Information Systems
The University of Melbourne, Parkville, Victoria, Australia
mariom@student.unimelb.edu.au

Abstract. Algorithm selection and configuration is a challenging problem in the continuous optimization domain. An approach to tackle this problem is to develop a model that links landscape analysis measures and algorithm parameters to performance. This model can be then used to predict algorithm performance when a new optimization problem is presented. In this paper, we investigate the use of a machine learning framework to build such a model. We demonstrate the effectiveness of our technique using CMA-ES as a representative algorithm and a feed-forward backpropagation neural network as the learning strategy. Our experimental results show that we can build sufficiently accurate predictions of an algorithm's expected performance. This information is used to rank the algorithm parameter settings based on the current problem instance, hence increasing the probability of selecting the best configuration for a new problem.

Keywords: Automatic analysis of algorithms, algorithm configuration, heuristic methods, randomized algorithms, meta-learning models.

1 Introduction

One of the most interesting questions in search and optimization is: "Before we perform a run, can we estimate the likelihood that the algorithm a will be successful on a given continuous optimization problem f?" In most circumstances, it is very difficult to answer this question. It is a well-known fact that each search algorithm exploits particular characteristics of the landscape [1]. As such, it is very optimistic to expect that an algorithm would work reasonably well across a wide range of continuous optimization problems unless some restrictions are in place [2]. However, it is possible to use machine learning techniques to elucidate information related to the effects of algorithm selection and/or parameter settings on similar problems; an approach known as *meta-learning*.

Leyton-Brown and co-workers [3] describe an automated algorithm selection method for boolean satisfiability problems. Their approach, based on methods proposed by Rice [4], employs supervised learning to build a model that can predict algorithm runtime. By comparing the estimated performance, it was possible to select the algorithm most likely

* This work has been partially funded through a 2012-2013 DAAD/Go8 Grant.

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 226–235, 2012.

to be suited to the task at hand. Related work was also reported in [1]. Smith-Miles [5] describe a similar meta-learning framework that can be used to develop automated algorithm selection and ranking models. More generally, Hoos [6] describes automated techniques applicable for algorithm selection and configuration for NP-hard problems.

In this paper, we use the meta-learning framework outlined above to build a prediction model of algorithm performance for continuous optimization problems. The model inputs include information about f – such as the dimension, target value and landscape features – and the parameter settings of a. The model output is the algorithm performance measured by the number of required function evaluations to find a solution. Information about a new problem f_n can then be used by the model to estimate the performance of a on f_n. To illustrate the efficacy of this approach, we model instances of the well-known Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) using a feed-forward backpropagation neural network (NN). The knowledge base used for training the model is obtained by processing data from the Comparing Continuous Optimization (COCO) set of benchmarks [7]. To validate the model, we use data from the CEC2005 set of benchmarks [8].

The remainder of the paper is organized as follows. In Section 2, we discuss background material related to the meta-learning algorithm selection framework used in this study. Section 3 describes the meta-learning model for continuous optimization problems in detail. Here, we describe the landscape features and the performance metric used as inputs and outputs. Section 4 describes the experimental procedure based on alternative CMA-ES parameter settings. Section 5 presents the results obtained from our experiments. Finally, Section 6 discusses the results and states the conclusions. Avenues for further work are also identified in this section.

2 Background

A continuous optimization problem is such that, given a cost function f that maps the search set $\mathcal{X} \subset \mathbb{R}^n$ to the objective set $\mathcal{Y} \subset \mathbb{R}$, we want to find one or more candidate solutions $\mathbf{x}_o \in \mathcal{X}, y_o = f(\mathbf{x}_o)$, such that $|y_o - y^\star| \ll \delta$, where $\delta \to 0$ and $\mathbf{x}^\star \in \mathbb{R}^n, y^\star = f(\mathbf{x}^\star)$ are the location of the global optimum and its value.

This type of problem is usually described through the search landscape metaphor [9]. A landscape $\mathcal{L}$ for a function f is defined as the tuple $\mathcal{L} = \{\mathcal{X}, f, d\}$, where d denotes a distance measure. The distance relates solutions among each other, hence it allows the systematic search for $\mathbf{x}_o$ in $\mathcal{X}$. Ideally, we expect that our search produces an acceptable solution after a bounded number of function evaluations. The opposite case is known as *premature convergence*, when the search is unable to generate solutions outside a small area under examination and the solution obtained is unacceptable.

As a direct consequence of the large number of existing optimization algorithms, it is difficult to determine which algorithm is able to efficiently exploit the search landscape structure for a given problem [10]. Deciding which algorithm to use is referred to as the "algorithm selection problem" by Rice [4]. In his seminal work, Rice defined four different sets: The *problem set*, $\mathcal{F}$, which contains functions that map $\mathcal{X}$ to $\mathcal{Y}$; the *algorithm set*, $\mathcal{A}$, which contains algorithms capable of searching for $\mathbf{x}_o$ in $\mathcal{X}$; the *performance set*, $\mathcal{P} \subset \mathbb{R}$, which contains the feasible values of $\rho(f, a)$, a measure for the

cost of applying an algorithm $a \in \mathcal{A}$ in a problem f; and the *set of landscape features*, $\mathcal{C} \subset \mathbb{R}^m$, which is a set of attributes of the functions in $\mathcal{F}$. These features are selected in a way such that varying complexities are exposed, known structural properties are captured, and any known advantages and limitations of the different algorithms can be related to the features.

The algorithm selection problem investigated by Rice has been extended and evaluated in a variety of computational problem domains using a meta-learning framework [1,3,5]. In this framework, algorithm selection is performed by exploring insights gained from previous experiments. Here, a function $g : \mathcal{C} \mapsto \mathcal{P}$ can be used to predict the algorithm performance based on specific input features. If we know beforehand the values of the features of a subset of functions from $\mathcal{F}$ and the values of ρ for an algorithm in $\mathcal{A}$; then, it is possible to use a learning strategy (such as linear regression) to identify the function g. When a new problem is encountered, g (or the performance model) can be used to predict the performance of the algorithm.

When working in the continuous optimization domain, careful attention must be given to the design of model inputs and outputs. Inputs to the model g may include the set of landscape features $\mathcal{C}$, and parameters of the algorithm set, $\mathcal{A}$. However, calculating the set of landscape features may well be a stumbling block, as this is a non-trivial computation [11, 12]. Previous work in the continuous optimization domain recognizes the necessity to link landscape features and algorithm parameters to algorithm performance [13–18]. Subsequently, an appropriate learning strategy must be employed to generate model output $\rho\,(f,a)$ based on a range of algorithm parameter values.

3 Prediction Model

Our model is an implementation of the meta-learning framework described above tailored for continuous optimization problems. Here, we build a regression model. By considering the landscape features and algorithm parameters as independent variables and algorithm performance as the dependent variable, the model can be used to predict algorithm behavior for a given problem. A high-level overview of the model is presented in Figure 1.

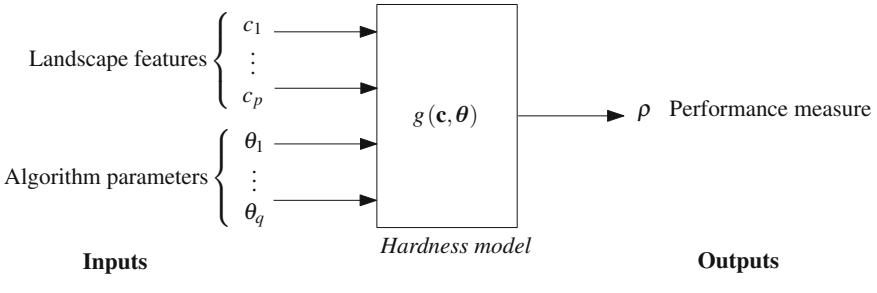

Fig. 1. Structure of a meta-learning model for a continuous optimization algorithm

3.1 Model Inputs

Landscape Features. Over the past few decades, when compared with the number of novel algorithms, relatively limited attention has been given to the question of what attributes a certain problem has, how to quantify them, and how the performance can be related to such attributes [18]. This is because, among other reasons, it is not a single attribute that defines the difficulty, but the interplay between different attributes [19]. Landscape analysis methods provide descriptive statistics related to algorithm performance. However, designing suitable analysis methods for a given domain is not straightforward. In some cases, the effort in calculating exact values of these statistics is greater than running a simple search algorithm [11, 12]. Approximations can be efficiently calculated, but the question remains if the loss of precision is too large to work with [20]. Besides, analyzing a whole landscape by using a single statistic is overly optimistic [18, 21, 22]. These are limitations that must be acknowledged.

For this work, we selected the following features:

- Dispersion (DISP) [23] identifies features of the global structure. It is defined as the pairwise distance between the q best points — usually $q = 100$ — from a sample of size p, as shown in (1).

$$DISP = \frac{1}{q\,(q-1)} \sum_{i=1}^{q} \sum_{j=1, j\neq i}^{q} d\,(\mathbf{x}_i, \mathbf{x}_j) \tag{1}$$

- Fitness distance correlation (FDC) [24] identifies the relationship between the position and the cost value, and has demonstrated capability to identify deceptiveness in the landscape. To calculate FDC, assume that from a sample of size p, $\hat{\mathbf{x}}_o = \arg\min f\,(\mathbf{x}_i)\,, i = 1, \ldots, p$ and $\hat{y}_o = f\,(\hat{\mathbf{x}}_o)$. Then, FDC is calculated using (2), where $d = \|\hat{\mathbf{x}}_o - \mathbf{x}_i|$, $\bar{y}$ and $\bar{d}$ are the averages of the cost and the distance, and $\hat{\sigma}_y$ and $\hat{\sigma}_d$ are the standard deviation of the cost and the distance.

$$FDC = \frac{1}{p-1} \sum_{i=1}^{p} \left(\frac{y_i - \bar{y}}{\hat{\sigma}_y} \right) \left(\frac{d_i - \bar{d}}{\hat{\sigma}_d} \right) \tag{2}$$

- Multiple correlation coefficient (R^2) identifies the relationship between n variables using a linear model approximation. Let R^2 be calculated using (3), where $\mathbf{r}_{\mathbf{x}y}$ is the vector of cross-correlations between the predictor variables on $\mathcal{X}$ and the criterion variable on $\mathcal{Y}$ and $\mathbf{R}_{\mathbf{xx}}$ is the matrix of inter-correlations between predictor variables.

$$R^2 = \mathbf{r}_{\mathbf{x}y}^{\top} \mathbf{R}_{\mathbf{xx}}^{-1} \mathbf{r}_{\mathbf{x}y} \tag{3}$$

- Variable significance [25] estimates the amount of information that a subset of predictor variables provides for the criterion variable. Let $\mathcal{V} = \{1, \ldots, n\}$ be a set of variables indexes, $v \in \mathcal{V}$ the index of one of such variables, and $V \subset \mathcal{V}$ be any combination of such indexes. The significance of V is calculated by (4), where $\hat{I}\,(\mathcal{X}_V; \mathcal{Y})$ is the estimated mutual information and $\hat{H}\,(\mathcal{Y})$ the estimated entropy of $\mathcal{Y}$. Let $\zeta^{(k)}$ and $\sigma_{\varepsilon}^{(k)}$ be the average significance of order k and its standard deviation respectively, where $k = |V|$.

$$\zeta(V) = \frac{\hat{I}(\mathcal{X}_V;\mathcal{Y})}{\hat{H}(\mathcal{Y})} \tag{4}$$

$$\zeta^{(k)} = \frac{1}{\binom{n}{k}} \sum_{V\subset\mathcal{V},|V|=k} \zeta(V) \tag{5}$$

$$\sigma_\zeta^{(k)} = \sqrt{\frac{1}{\binom{n}{k}} \sum_{V\subset\mathcal{V},|V|=k} \left(\zeta(V) - \zeta^{(k)}\right)^2} \tag{6}$$

– Entropic epistasis [25] evaluates the contribution that a single variable has to the fitness given the state of other variables. For a variable subset V the entropic epistasis is calculated by (7). Let $\varepsilon^{(k)}$ and $\sigma_\varepsilon^{(k)}$ be the average entropic epistasis of order k and its standard deviation.

$$\varepsilon(V) = \frac{\hat{I}(\mathcal{X}_V;\mathcal{Y}) - \sum_{v\in V} \hat{I}(\mathcal{X}_v;\mathcal{Y})}{\hat{I}(\mathcal{X}_V;\mathcal{Y})} \tag{7}$$

$$\varepsilon^{(k)} = \frac{1}{\binom{n}{k}} \sum_{V\subset\mathcal{V},|V|=k} \varepsilon(V) \tag{8}$$

$$\sigma_\varepsilon^{(k)} = \sqrt{\frac{1}{\binom{n}{k}} \sum_{V\subset\mathcal{V},|V|=k} \left(\varepsilon(V) - \varepsilon^{(k)}\right)^2} \tag{9}$$

Algorithm Parameters. The "algorithm selection" problem in many cases is equivalent to the "algorithm configuration" problem as it is possible to consider two instances of the same algorithm as two completely different ones if they differ only in one parameter [26]. An experimentally driven meta-learning approach has been suggested for the latter problem [12]. Thus, we adopt this approach in our model.

We use CMA-ES as the base algorithm for our investigation, and the following parameters as inputs for the model.

– Target precision (e_{target}) is the error between the best solution and the target solution. As many practical problems do not have a target solution, a value can be developed if we consider an improvement by certain user defined percentage over the best known solution before the experimental run.
– Size of the population (λ).
– Depending on the algorithm, it might be possible to have rules that define how the individuals are generated or evaluated. Mirrored ($M \in \{0,1\}$) indicates whether the offspring are generated in pairs, where one is 180° from the other. Serialized ($S \in \{0,1\}$) indicates if the offspring is evaluated sequentially. Hence, if an improving offspring is found the others are not evaluated.

3.2 Model Output

The performance ρ of the model is set to be the expected running time $\hat{t}$ of the algorithm. Here, $\hat{t}$ provides an estimation of the average number of function evaluations required by the algorithm a to reach y_{target} for the first time [7]. The expected running time is calculated as follows:

$$\hat{t}(f,a,y_{\text{target}}) = \frac{\#\text{FEs}\left(y_{\text{best}} \geq y_{\text{target}}\right)}{\#\text{succ}} \tag{10}$$

where $\#\text{FEs}\left(y_{\text{best}} \geq y_{\text{target}}\right)$ is the number of function evaluations over all trials where the best function value, y_{best}, was not smaller than the target function value, and #succ is the number of runs where target precision was achieved. When not all trials are successful, $\hat{t}$ depends strongly on the termination criteria of the algorithm.

3.3 Regression Model

In this study, we use an NN to build the model. It is important to note that the meta-learning framework is flexible and any appropriate learning strategy could be used. While other regression methods might provide different — even superior — accuracy, as a proof of concept the NN will suffice.[1]

To train this model, inputs include landscape features (from a collections of problem instances of various complexities) and algorithm parameter values (corresponding to alternative instantiations of the CMA-ES algorithm) as described in Section 3.1. The model output value is $\log_{10}(\hat{t})$. Since the target precision e_{target} can take different values for the same problem, a pattern for each e_{target} is created where the other input values are kept constant.

The accuracy of the resulting model depends on several factors: the diversity in the knowledge base used to train the model, the relevance of the features and their precision, and the training method used in the model. For our purpose, the accuracy of the model would be evaluated on the capability to provide a realistic ranking of the different configurations of the algorithm. While an accurate estimation of the expected running time would be desirable, at this exploratory stage it is unlikely to be obtained.

4 Experiments

A comprehensive set of simulation experiments were performed to evaluate the efficacy of the proposed meta-learning prediction model of algorithm performance for continuous optimization problems.

A multi-layered feed-forward neural network (2 hidden layers; 10 neurons in each layer) was used for the regression model. The training method employed was the Levenberg-Marquardt back-propagation algorithm. The inputs and outputs of the model were normalized in the $[-1,1]$ range. MATLAB version 2009b was used for implementation.

[1] We leave the evaluation of alternative learning strategies to future work.

Eight configurations of the CMA-ES algorithm were used for this experiment: standard, mirrored, serialized and mirrored-serialized with a single parent and either two or four offspring.

To train the model, we use the functions from the COCO noiseless set [7] in $\{2,3,5,10,20\}$ dimensions. The features are calculated using fifteen runs of $10^3 \cdot n$ function evaluations using uniform random sampling ($p = 15 \cdot 10^3 \cdot n$). While it seems that a large amount of data must be collected before the model can be used, we presume that it is possible to use data from an algorithm run to calculate the features, hence avoiding the need for an additional data extraction experiment. However, this hypothesis is not tested in this work. The performance metric is calculated over fifteen runs for each of the eight CMA-ES configurations with a target value of 10^{-8} and a maximum number of function evaluations equal to $10^4 \cdot n$.

To evaluate the effectiveness of our model in predicting the performance of a given algorithm on new problems, we use a subset of the problems from the CEC2005 benchmarks [8]. We limit the evaluation of our model to the noiseless functions and those functions whose optimum is inside the initialization region. The functions were evaluated on $\{2,3,5,10,20\}$ dimensions. The collection of metric values for the CEC2005 problems followed the same procedure as for COCO benchmark functions.

5 Results

We examine the predictions made by our model using the CEC benchmarks. The model is fed with landscape features that represent a benchmark problem ($\mathbf{c}$), algorithm parameters that define the configuration ($\boldsymbol{\theta}$), and a desired target precision (e_{target}). Then, the predicted $\hat{t}$ for each configuration is ranked from the lowest to the highest at a fixed e_{target}. Figures 2(a) and 2(b) show the resulting rankings for the 5-dimensional versions of the Sphere function and the Hybrid composition function 1, respectively. The top plots represent the actual ranking while the bottom plots represent the predicted ranking. The abscissa are the $\log_{10}\left(e_{\text{target}}\right)$ organized from lowest precision on the left to the highest precision on the right. The ordinates are the ranking of a given configuration, where the lowest is the worst performing and the highest is the best performing. Each line on the plot represents a configuration.

We quantify the similarity between rankings using the following procedure. For a fixed e_{target}, let r_a be the ranking based on the actual performance of each configuration and r_p be the ranking produced by sorting the predictions generated by the model. Let $\delta_p = \left|r_a - r_p\right|$ be the difference between the two rankings. When this measure is calculated for CEC benchmark data, the average δ_p over all scenarios — for each target precision, dimension and benchmark function — is 12.69.

A baseline is required to asses the impact of the differences in rankings. Let r_r be a random ranking — the positions in the ranking of each configuration are randomly generated — which is kept fixed for all the scenarios. Let δ_r be the difference between the random to the actual ranking. We produce 100 different random rankings, each with its own value of δ_r. The average value of the δ_r is 21.01, which is 39.59% higher than the average δ_p. This indicates that by using the predicted ranking we improve our chances to select the best algorithms early on the experiment. Table 1 lists the difference between δ_r

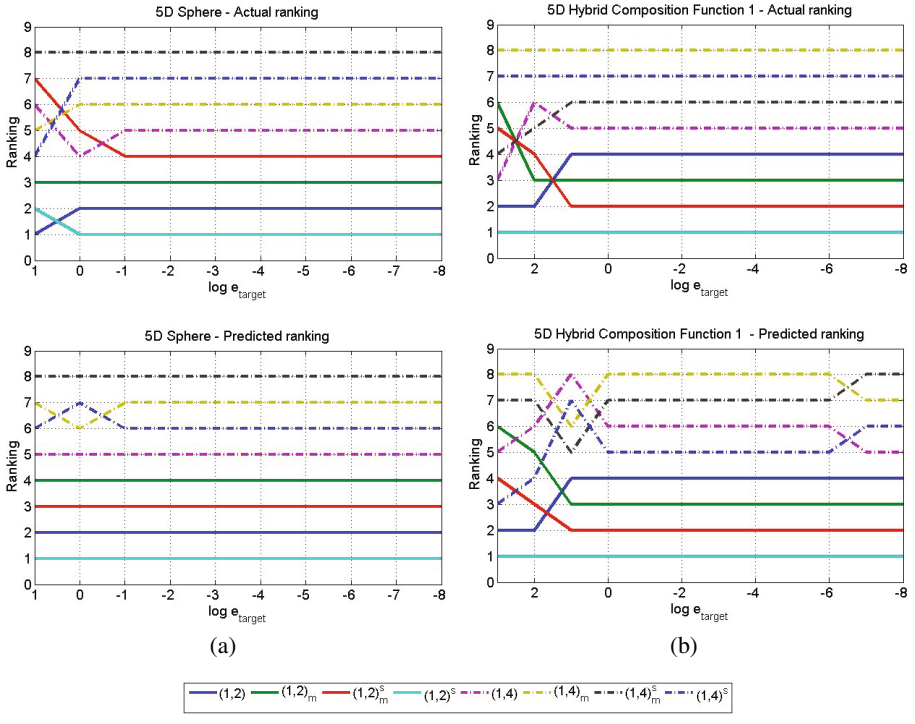

Fig. 2. Actual and predicted rankings for the different CMA-ES configurations on the 5D Sphere function (a) and on the 5D first hybrid composition function (b) of the CEC2005 benchmark

Table 1. Average difference between the random and predicted rankings. A negative value indicates a scenario where the random ranking is more accurate than the predicted ranking.

f	2D	3D	5D	10D	20D	f	2D	3D	5D	10D	20D
1	7.42	8.85	14.28	7.81	4.59	12	4.17	6.19	9.54	6.03	6.76
2	8.11	8.17	13.14	8.37	4.41	13	15.26	8.72	-2.54	-10.74	21.48
3	10.33	5.89	11.37	8.58	10.73	14	5.07	7.88	13.90	6.48	2.94
4	12.28	5.55	10.74	14.06	-0.16	15	12.96	9.09	6.23	10.40	10.55
5	8.36	9.42	12.65	8.40	8.45	16	4.42	3.67	4.21	2.52	1.12
6	6.68	11.23	6.94	3.78	3.09	17	3.21	6.30	8.10	4.90	4.88
7	4.31	11.49	16.18	8.63	20.14	18	11.64	18.11	18.34	10.19	12.08
8	9.88	11.19	15.10	12.64	7.20	19	2.41	-5.44	3.20	4.19	1.34
9	9.43	5.59	16.68	8.60	12.62	20	-2.73	10.02	14.90	1.06	1.36
10	7.94	8.53	10.97	11.61	4.58	21	11.37	15.79	5.27	2.79	-5.13
11	5.39	3.35	10.41	6.63	-3.22						

and δ_p for each benchmark function when averaged over e_{target}. With some exceptions on f_4, f_{11}, f_{13}, f_{19}, f_{20} and f_{21}, in most cases the dissimilarity between δ_p is lower than δ_r. When this is not the case, the differences can be due to the precision of the model, which can be improved by changing the learning strategy or preprocessing the input data.

6 Conclusion

In this paper, we used a meta-learning framework to build a model that captures the relationship between problem cost function structure, algorithm parameters and a given performance metric. The model is used to predict algorithm performance measured in terms of the number of function evaluation required.

A NN was used as the underlying learning strategy. The network inputs included a number of landscape characteristics (calculated using well-known statistical estimators) and selected parameter settings. The network was trained, and subsequently tested, using a suite of benchmark continuous optimization problems with varying characteristics. The simulation results clearly demonstrate that the model was able to predict the relative ranking values for given algorithm-parameter combinations effectively.

Model performance was measured by comparing predicted and actual rankings of algorithm parameter settings on new problem instances. This implies that the ranking can be verified if all the configurations are tested. In a practical situation this is often not the case, as the experiments will be censored when an acceptable solution is reached. Examining the effects of censoring on the rankings is one avenue of future work.

All landscape analysis was performed off-line — an initial experiment was carried out to calculate landscape statistics that form part of the input of our model. In future work, we will transfer such calculations to an on-line mode. The benefits of such an approach are highlighted by reviewing Figures 2(a) and 2(b). Note that the best configuration is not the same at all target values. An on-line prediction mechanism coupled with the ability to switch between configurations offers the potential for significant improvement in optimization performance.

References

1. Hutter, F., Hamadi, Y., Hoos, H.H., Leyton-Brown, K.: Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 213–228. Springer, Heidelberg (2006)
2. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1(1), 67–82 (1997)
3. Leyton-Brown, K., Nudelman, E., Shoham, Y.: Empirical hardness models: Methodology and a case study on combinatorial auctions. J. ACM 56, 22:1–22:52 (2009)
4. Rice, J.: The algorithm selection problem. In: Advances in Computers, vol.15, pp. 65–118. Elsevier (1976)
5. Smith-Miles, K.A., James, R.J.W., Giffin, J.W., Tu, Y.: A Knowledge Discovery Approach to Understanding Relationships between Scheduling Problem Structure and Heuristic Performance. In: Stützle, T. (ed.) LION 3. LNCS, vol. 5851, pp. 89–103. Springer, Heidelberg (2009)
6. Hoos, H.H.: Programming by optimization. Commun. ACM 55(2), 70–80 (2012)
7. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking BBOB-2010: Experimental setup. Technical Report RR-7215, INRIA (September 2010)

8. Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report, NTU, Singapore and IIT, Kanpur (2005)
9. Reeves, C.: Fitness landscapes. In: Search Methodologies, pp. 587–610. Springer (2005)
10. Hough, P., Williams, P.: Modern machine learning for automatic optimization algorithm selection. In: Proceedings of the INFORMS Artificial Intelligence and Data Mining Workshop (2006)
11. He, J., Reeves, C., Witt, C., Yao, X.: A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability. Evol. Comput. 15(4), 435–443 (2007)
12. Smith-Miles, K.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. 41(1), 6:1–6:25 (2009)
13. Francois, O., Lavergne, C.: Design of evolutionary algorithms-a statistical perspective. IEEE Trans. Evol. Comput. 5(2), 129–148 (2001)
14. Steer, K.C.B., Wirth, A., Halgamuge, S.K.: Information Theoretic Classification of Problems for Metaheuristics. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 319–328. Springer, Heidelberg (2008)
15. Malan, K., Engelbrecht, A.: Quantifying ruggedness of continuous landscapes using entropy. In: Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC 2009), pp. 1440–1447 (May 2009)
16. Caamaño, P., Prieto, A., Becerra, J.A., Bellas, F., Duro, R.J.: Real-Valued Multimodal Fitness Landscape Characterization for Evolution. In: Wong, K.W., Mendis, B.S.U., Bouzerdoum, A. (eds.) ICONIP 2010, Part I. LNCS, vol. 6443, pp. 567–574. Springer, Heidelberg (2010)
17. Mersmann, O., Preuss, M., Trautmann, H.: Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 73–82. Springer, Heidelberg (2010)
18. Müller, C.L., Sbalzarini, I.F.: Global Characterization of the CEC 2005 Fitness Landscapes Using Fitness-Distance Analysis. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A.I., Merelo, J.J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G.N. (eds.) EvoApplications 2011, Part I. LNCS, vol. 6624, pp. 294–303. Springer, Heidelberg (2011)
19. Richter, H.: Coupled map lattices as spatio-temporal fitness functions: Landscape measures and evolutionary optimization. Phys. Nonlinear Phenom. 237(2), 167–186 (2008)
20. Watson, J., Howe, A.: Focusing on the individual: Why we need new empirical methods for characterizing problem difficulty. In: Working Notes of ECAI 2000 Workshop on Empirical Methods in Artificial Intelligence (August 2000)
21. Beck, J., Watson, J.: Adaptive search algorithms and fitness-distance correlation. In: Proceedings of the Fifth Metaheuristics International Conference (2003)
22. Smith, T., Husbands, P., Layzell, P., O'Shea, M.: Fitness landscapes and evolvability. Evol. Comput. 10(1), 1–34 (2002)
23. Lunacek, M., Whitley, D.: The dispersion metric and the CMA evolution strategy. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 477–484. ACM, New York (2006)
24. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 184–192. Morgan Kaufmann Publishers Inc. (1995)
25. Seo, D., Moon, B.: An information-theoretic analysis on the interactions of variables in combinatorial optimization problems. Evol. Comput. 15(2), 169–198 (2007)
26. Rice, J.: Methodology for the algorithm selection problem. In: Proceedings of the IFIP TC 2.5 Working Conference on Performance Evaluation of Numerical Software (1979)