EA-based Parameter Tuning of Multimodal Optimization Performance by Means of Different Surrogate Models

Catalin Stoean
University of Craiova
Romania
catalin.stoean@inf.ucv.ro

Mike Preuss
TU Dortmund
Germany
mike.preuss@tu-dortmund.de

Ruxandra Stoean
University of Craiova
Romania
ruxandra.stoean@inf.ucv.ro

ABSTRACT

In the current study, parameter tuning is performed for two evolutionary optimization techniques, Covariance Matrix Adaptation Evolution Strategy and Topological Species Conservation. They are applied for three multimodal benchmark functions with various properties and several outputs are considered. A data set with input parameters and metaheuristic outcomes is used for training four surrogate models. They are then each used by a genetic algorithm that is employed for searching the best parameter settings for the initial approaches. The genetic algorithm uses the model outputs as the direct fitness evaluation and only the best found parameter setting is tested within the original metaheuristics. Each model quality is priory evaluated, but they are all subsequently used in the search process to observe how the (in)accuracy influences the final result. Additionally, the genetic algorithm is used for tuning these approaches directly to test if the search conducts to the same parameter set, or at least close to it.

Categories and Subject Descriptors

G.1.6 [**Mathematics of Computing**]: Optimization—*Global Optimization, Unconstrained Optimization*;
I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge Acquisition*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Experimentation, Measurement, Performance

Keywords

parameter tuning, surrogate modeling, function optimization, evolutionary algorithms

1. INTRODUCTION

Although evolutionary algorithms (EAs) have become widely acknowledged tools for solving real-world optimization problems, a step back still exists from some specialists in industry who claim that they need too many objective function evaluations to reach acceptable results. In industry, many evaluations often mean expensive simulations with respect to money and/or time. Usually, the acceptable solution is to employ a meta-model that learns from a set of parameter values used in previous simulations together with their results and is able to predict the output for new settings in order to provide an anticipated fitness evaluation for the EA optimizer.

But how good are such surrogate models, especially when their output cannot be checked during the evolutionary process, how good is the re-created landscape? It often happens that the meta-model possesses optima that do not coincide with those of the original fitness function. There are several possibilities that can be explored in order to establish a *model management*, as referred to in conventional optimization [13], or *evolution control*, as in the evolutionary computation (EC) community [15]. However, in this work we focus on the simple but frequently used in practice models that learn from history data, their predicted output being used directly as fitness evaluation in the optimizer, with only the final best solution being validated on the original optimization function (recommendation systems where the best solution on the model is returned as recommendation).

The overall task of this work is to detect which measures are to be used in a tuning process based on landscape models if we want to attain good performance in multimodal optimization, and to evaluate the suitability of different types

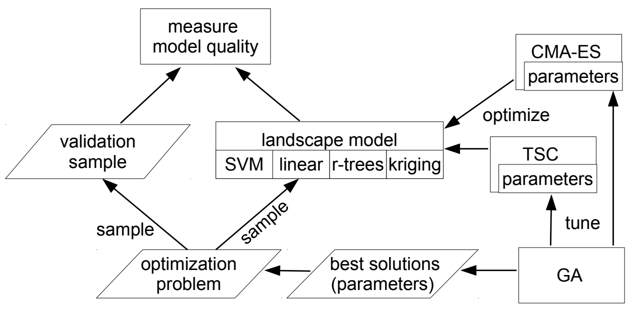

Figure 1: A GA tunes the two optimization algorithms on a landscape model build from samples of the original function, employing different models.

of models for this process. The employed algorithms can be rather considered examples, others could be used instead. The 'big picture' is also documented in Fig. 1.

The approaches we will tune here are the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [11] in its IPOP variant and Topological Species Conservation (TSC2) [24], we address different numbers of tuned parameters, while four different meta-models will be employed, and their results will be compared to the ones of the original function used as fitness criterion. While TSC2 was especially designed for tackling multimodal problems and does so in a parallel fashion (population is spread over the interesting regions), the IPOP CMA-ES approaches such problems by population size adaptation and frequent restarts. In order to be comparable to TSC2, the best attained solutions throughout a run are collected and the result used for measuring.

The paper is organized as follows: Next section briefly presents a survey on using surrogate models in EC, section 3 points out some generalities within parameter tuning and describes the mechanics behind the approaches that will be furthered experimentally compared in section 4. Section 5 provides some conclusions and ideas to be further investigated.

2. SURROGATE MODELS IN EC

Naturally, the most accurate manner to observe the behavior of a model under different settings is to actually test that model under the desired scenarios. However, as that is not always possible (expensive simulations may be necessary), approximations of the model outcome are used instead. These approximations are obtained by replacing the actual model with a surrogate one that can be fed with the same initial settings and, hopefully, leads to similar outcomes as the original.

In EC, the output of the surrogate model is usually considered as the direct fitness evaluation of the individual that encompasses characteristics of the initial model; other approaches, like using it only for migration in island architectures, or for mutation and recombination, have also been proposed, but we will focus herein only on the direct evaluation situation - for more details about the other approaches, please see [13]. For a surrogate model to have the results close to that of the original, there are several control techniques that can be applied:

Individual-based : In each generation, some individuals obtain the approximated evaluations, while for the rest original ones are generated. Surely, the aim is to reduce as much as possible the number of individual evaluations using the original model in order to limit the cost, but at the same time have an accurate model. The question that arises is how to efficiently choose the individuals to be evaluated by the original model: the fittest ones, and the higher the approximation accuracy, the more often approximations are used instead [15], or the prototypes from the population separated into clusters [16], or simply pick some random individuals [14].

Generation-based : Once in a fixed number of generations, the entire population is evaluated using the original function or only the population from the final generation [20].

No evolution control : Most of the time, there is no control at all, meaning that only the approximation of the function is used as individual fitness during the entire evolution process.

Among the surrogate models that are usually employed for approximating the functions, there are the *polynomial* ones, also known as response surface models, *Gaussian*, also known as Kriging in traditional design optimization, *support vector machines* (SVMs) or *neural networks*. For further reading about them, please see [13].

There are several papers that compare the performance of different approximation models [4], [5], [10], [12], [22], [23]. Usually the benchmark problems are represented by functions and some of these papers do not employ an EA to test where the model might converge, but only compute the accuracies of the surrogate models for a number of confirmation points, as the sum of absolute differences between the approximated and the original evaluations. Among them, there are also many interesting studies that are focused on various effects of evolution control and on the behavior of neural networks as models.

In these circumstances, our aim is to conduct a comparative study over several surrogate models for the hot problem of tuning the parameters of two approaches that are applied for function optimization. So, the challenge comes from the fact that we do not try to reconstruct the landscape of the function to be optimized, but we intend to mimic the parameter behavior of two evolutionary methods (that are stochastic and provide noisy outcomes) when they deal with several intricate, low and high-dimensional, multimodal optimization functions.

3. TUNING METAHEURISTICS

Traditionally, parameter setting has been performed manually, although ways for automatic regulation have been reported since the 1980s. Nevertheless, recent years have demanded an aggressive shift from the time-wasting, precision-susceptible human adjustment of parameters to computationally sustained, rapid estimation of their appropriate values for success. Usually, parameter adjustment is performed using a direct evaluation for the threshold variable values within the tuned model.

3.1 State of the Art

Parameter setting within any metaheuristic is an issue of essential importance that significantly accounts for the performance of the respective algorithm. Finding the proper values for the involved parameters in order to make a method perform well is imperative from several perspectives:

- The inner-workings of any technique depend on an appropriate choice for its intrinsic variables.
- Metaheuristics usually exhibit an intricate design and/or a large number of parameters. Both target a great computational effort in adjusting many variables possibly within a complex working scheme.
- There is a big number of options for the value of each possible variable and generally little knowledge on the effect these values have on the maintenance of equilibrium between exploration and exploitation of the search space, which ultimately lead to the ability of

the metaheuristic to converge to the (near) optimal solution.

- The parameters strongly depend on each problem instance that is currently at hand.

- The use of rough estimations for some parameter values that are given by different theoretically proven formulas do not always lead to the (near) optimum or have all the necessary knowledge on the problem to be accurate enough.

There have been numerous attempts to tackle the automated setting of parameters that control the behavior of a metaheuristic. These can fall into two categories of approaches for selecting parameter values [8]:

- *Parameter control* (or *online*), where the values are changed during the run of the algorithm. It has been applied for various metaheuristics like Evolution Strategies [17], Genetic Algorithms (GA) [9], Particle Swarm Optimization [3], Differential Evolution [19], Ant Colony Optimization [18].

- *Parameter tuning* (*offline*), where the values for the parameters are chosen a priori and do not suffer changes during the run. These methods can be non-iterative, where the various values are generated only once for the parameters to be set, and iterative, where there is a small initial set of obtained configurations and then, based on the results following these configurations, some iterative steps are followed to exploit the previous findings and explore the more promising regions of the search space.

Interestingly, there are border situations where tuning can be realized within one run of an optimization algorithm and effectively is then a parameter control method [26].

We focus on the iterative methods within parameter tuning and, more precisely, on using surrogate models for parameter tuning. A previous such approach is introduced by [6], where a linear regression model is learned in a first stage from a set of configurations generated over the parameter domains and then, in a second stage, a local search procedure is used to generate new vectors based only on the model approximation. A well-known framework, sequential parameter optimization (SPO) [1], also contains a multi-stage procedure that involves models: It starts with a set of configurations generated over the search space, they are evaluated using the original approach to be tuned and a model (Kriging, in this case) is used to generate new configurations with a high utility that are then reevaluated using the original algorithm to be tuned.

3.2 Parameter Tuning via Fitness Approximation

We follow the direction opened by Coy [6], but employ several models (four, plus the original method to be tuned) and combine them with a simple GA as tuning method. We tune two algorithms, each with a different number of parameters to be set, in order to optimize three functions with various characteristics.

The tuning process follows the steps in Algorithm 1. The metaheuristics that are selected to be tuned are the CMA-ES [11] and TSC2 [24]. While for the first we choose three parameters for tuning, for the latter we select five. As the evolution process is based only on the outputs of the learned model, it is important to measure how accurate that model is on a test set that is completely different from the collection used for training (line 4). For further reading regarding the importance of the meta-model validation, please see [2]. More details regarding the settings of the algorithm in our implementation are discussed in the next section.

Algorithm 1 Parameter tuning via a surrogate model

```
Require: Metaheuristic, definition intervals for the parameters
    to be tuned and a model to resemble the original algorithm
    behavior
Ensure: Specific values for the parameters
1: (Randomly) generate a set of vectors with input values inside
   the definition intervals for the parameters to be tuned
2: Apply the metaheuristic on the samples and provide one output
   for each configuration
3: Apply a regression model to learn from this formed training
   set the correspondence between the input values and meta-
   heuristic output
4: Validate the model on a set of test samples
5: Run a GA to find the best configuration using as fitness the
   outputs of the model on the individual values
```

The following models are selected as surrogates for the metaheuristics: linear regression, in order to have the same model as in the article that opened this research direction [6], SVMs with a radial kernel, Kriging and regression trees as models that are widely used for regression. Besides these models, we also used the actual metaheuristic to be tuned by the GA to check if the found configuration is similar to (any of) those found by the models on the one hand, and on the other, if the configurations are different, to see if the output also differs by a high amount. While we only have three functions for optimization, there are several outputs that are checked for these problems:

- *Number of peaks found* represents the number of detected peaks; a peak is considered *found* when an individual has the fitness distance to that of the desired optimum below 10^{-1}.

- *Peak accuracy* for one optimum is computed as the absolute difference in fitness value between the peak and the nearest individual in the population. When the function has several optima, these absolute difference values are summed.

- *Distance accuracy* refers to the dissimilarity in the genotypic space between each peak and its closest individual. It is computed in the same manner as peak accuracy, with the only change that the difference between fitness values is substituted by the Euclidean distance between the two individuals. It is very valuable when the test function possesses several optima with the same fitness value, as the peak accuracy can lead to inaccurate results.

- *Best fitness* represents the best fitness value found for an individual. It is used only in the case when there are several global optima, to eliminate erroneous results from the peak accuracy. Best fitness is computed from the population only, it is not related to the desired peaks at all.

However, not all outputs are considered for all problems because of two possible situations: In the initial configurations, for some outputs the metaheuristics always fail, while for others they always succeed, so the models could not learn any useful behavior from such results.

4. EXPERIMENTAL COMPARISON

4.1 Test Functions

We consider 2 functions with 2 variables, i.e. Six-hump camel back ($F1$) and Waves ($F2$): The former has 6 optima (2 global), equal two by two, that reside on a very smooth surface, while the latter has 10 optima (1 global) to be found, is asymmetric and some peaks are difficult to find as they lie on the border or on flat hills. The last function is considered for 10 variables: it is very rugged and is obtained as the result of the combination of several functions ($F3$), it contains 8 global optima and many local ones. Its definition is too large to be included here, however it is part of the collection of test cases that are used as benchmarks for the Congress on Evolutionary Computation 2013 Competition on Niching Methods and more information about it, including encoding, can be found in [27]. The definitions for the described functions are listed below.

- $F1(x,y) = -((4-2.1x^2+\frac{x^4}{3})x^2+xy+(-4+4y^2)y^2)$, $-1.9 \leq x \leq 1.9, -1.1 \leq y \leq 1.1$
- $F2(x,y) = (0.3x)^3-(y^2-4.5y^2)xy-4.7cos(3x-y^2(2+x))sin(2.5\Pi x))$, $-0.9 \leq x \leq 1.2, -1.2 \leq y \leq 1.2$
- $F3$ corresponds to $CF4$ in [27], $-5 \leq x_i \leq 5$

4.2 Task

Employ various surrogate models to learn the behavior of two different metaheuristics (CMA-ES and TSC2), one at a time, from a small amount of data that contains input parameters for CMA-ES and TSC2 and their original outputs, then use them to predict the outcome for other parameter settings evolved via a GA.

4.3 Experimental Setup

The parameters for CMA-ES and the considered intervals where threshold values are searched for are the following:

- The *factor* for increasing the population after a restart is considered for all functions in the interval $[1,2]$.
- The initial standard deviation is considered for $F1$ and $F2$ in the interval [0, 3], while for $F3$ in $[0,10]$.
- Population size (λ) is taken from $[3,100]$ in all cases.

For TSC2, we consider the following 5 parameters:

- Population size within the interval $[2,200]$.
- Recombination and mutation probabilities, each within the interval $[0,1]$.
- Mutation strength is considered for $F1$ and $F2$ in the interval [0, 3], while for $F3$ it is taken from $[0,30]$.
- The number of gradations, taken from $[1,15]$.

There are some significant differences between the intervals considered for drawing parameters of the two metaheuristics and they are further explained. The upper limit for the population size for CMA-ES is set to 100, while for TSC2 to 200, because within the former the $factor$ increases the number of candidate solutions with each restart and in some late evolution steps the actual population size can have a value much higher than 200. The intervals for the standard deviation within CMA-ES and mutation strength for TSC2 differ for the last function: While for the CMA-ES, high values conduct the search outside of the feasible solutions interval and the run stagnates, for TSC2 these large values enable escaping from local optima.

While for the readers not familiar with TSC2 the meaning of the first 4 parameters considered for tuning is quite obvious, the last one needs some explanations: It is used to check whether two different candidate solutions follow the same peak or not, by creating as many intermediary points between them as $gradations$ are appointed and by evaluating all these points to check if the fitness is also between those of the initial two points. When a generated solution does not have the fitness between the initial two, it is concluded that they lie in different basins of attracdtion. In conclusion, the higher the number of $gradations$, the more accurate the multimodality detection method is, but at a higher cost as concerns the number of consumed fitness evaluations.

For each metaheuristic and for each benchmark function, we established the same limit as concerns the number of samples used to train the models. We constructed a data set with 100 samples where each contains values for the enumerated parameters that are randomly generated in the given intervals. Each such sample of parameter values is consequently fed to the metaheuristic and all outputs mentioned in subsection 3.2 are computed and averaged over 10 repeats. Another similar data set with 50 randomly generated samples is created and kept only for model validation. The models are then learnt from the training set containing the 100 samples and used for predicting further outputs for new settings.

The stop condition is identical for CMA-ES and TSC2 applied to any function and refers to a budget of 10^4 fitness evaluations. Our intention is to get parameter settings as good as possible out of the two methods under the same limitations. One could argue that CMA-ES has an advantage over TSC2 since the former has only 3 parameters considered for tuning and the latter 5, while the same number of training samples are considered, so the models for the former will be more accurate. To answer that, we claim that the intention is not to compare the optimization power of the two metaheuristics, but rather observe how surrogate models used for the chosen task of parameter tuning act for different number of parameters under the same restrictions.

A GA with individuals that encode parameter settings for the current metaheuristic is evolved. The fitness evaluation is always returned by the employed regression model. As concerns the operators used within the GA, the choices are the commonly employed binary tournament selection, mutation with normal perturbation and intermediary recombination. The GA parameter values are a population size of 50, recombination probability of 0.4, mutation probability of 0.3 and mutation strength is set at the 10th part of each gene interval size. The intervals for the genes are bounded in the same manner as for the model training data and the stop condition is set to 2000 evaluations. Only the final best individual is validated on the original metaheuristic by actually running the approaches with the found parameter values. No matter what the current outcome used by the GA is, when using the final parameter set within the original approach, all the performance criteria are computed and are subsequently reported in the next subsection figures.

The encoding for the 4 models are used from the following R functions and packages: SVMs from package e1071 [7], the linear model from the widely used *lm* from the *stats* package, regression trees from package *rpart* [25] and Kriging from package DiceKriging [21]. They are employed with default values. A radial kernel is used for SVM (a linear one proved to be less efficient in pre-experimental trials), while for Kriging we considered *Universal Kriging* ("UK").

We measure the quality of the models prior to using them for providing fitness values to the GA. For that, the data set with 50 samples is used and the following measures are computed:

- The root mean square error (RMSE), computed as in (1), where n is the number of samples (50 in our particular case), P_i is the predicted outcome for sample i and A_i is its actual outcome.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i - A_i)^2} \qquad (1)$$

- Pearson correlation coefficient to measure how well the exact output values between model and validation match.

- Spearman's rank correlation coefficient which is computed as the Pearson correlation coefficient between the ranked variables.

4.4 Results and Visualization

Table 1 shows the results obtained for the 4 models when their prediction accuracy is tested against new data. For each function, we consider all models and compute only the outputs that are relevant. The last 5 lines contain the summed values (normalized in the case of RMSE) for each of the 4 models over all functions and measures. Figures 2 and 3 present the actual values of the obtained parameters for the best configurations found by the GA. Note that they are grouped by the performance criterion chosen for the tuning process (rows) and the modeled test problem (columns). The first row in each plot shows the model performance, that is the best non-validated performance found within tuning, measured with the chosen criterion. These values shall be compared to the appropriate row further down. These rows show the considered performance criteria (da = distance accuracy, pa = peak accuracy, ap = average number of peaks, bf = best fitness) and these results are obtained by appointing the found parameter set to the tuned approach, so they are validated results; this in opposition to the model line which, as previously stated, is the value returned by the model for the parameter set. The last 3 rows in each plot from Figure 2 and last 5 rows in each plot from Figure 3 represent the values for the parameters of the two approaches.

4.5 Observations and Discussion

In Table 1, for $F1$ and $F2$ outputs like the average number of peaks detected, average peak accuracy and average distance accuracy are considered for both metaheuristics; best fitness is not considered because both CMA-ES and TSC2 achieve very good results for this output for a very wide area of parameter settings, and it cannot measure how well the algorithms are suited to find multiple optima.

Concerning $F3$, peak accuracy is omitted for both metaheuristics because it is very misleading: the population for each of the two methods gathers on a peak and, especially if that optimum is a global one, the overall peak accuracy (for all 8 global optima) will return a very good value in this case. Even if an individual is marked as the one that conquered the closest peak, a different one also has an evaluation that is very close to that of the first and so on, for all the desired global peaks, peak accuracy becomes very good. On the other hand, distance accuracy is a great alternative to this measure and it is used for both metaheuristics.

From what we observed during the experiments, TSC2 is a method that requires more fitness evaluations to reach any of the global optima of $F3$. On the other hand, CMA-ES reaches exactly one peak for a wide range of parameter settings. One could say that both methods are already very good in the situations they have been designed for, TSC2 for detecting multiple peaks, and CMA-ES for reaching a good peak very quickly. Hence, we decided to take different outputs for the two metaheuristics on $F3$: while for CMA-ES the interest is in the number of peaks found, for TSC2 the best fitness comes as a more appropriate choice.

It is interesting to observe in the same Table 1 that there can be cases when the Pearson correlation coefficient is higher than the Spearman one and vice versa. The explanation for the first case is that the values for the model are very close to the one of the original approach, but their ranking is not the same, while the latter case occurs when the ranking is more often similar, but the differences in values are higher.

In Table 1, while for RMSE lower values are better, for the correlation coefficients values closer to 1 are preferred (1 meaning direct correlation, 0 is no correlation and -1 inverse correlation). The last 5 lines in Table 1 contain the summed values corresponding to each model. Since for the RMSE, the intervals differ greatly from one function to another and from one output to a different one, we decided to first normalize the values to the interval [0, 1] for each function and measurement and we only then summed these obtained normalized values. The best values are written in bold: the Kriging model appears to be the most accurate, followed by SVMs and regression trees which are very close to each other and, finally, linear regression.

There exist certain situations when some models perform very poorly, while others are very accurate, e.g. see for number of peaks found for TSC2, both Spearman and Pearson correlation coefficients for $F1$, where only the linear model is very imprecise. However, the accuracy of the models is not performed with the aim to decide which models are next selected for tuning via GA; all models are used as fitness providers for the GA as we intend to check how good the final found parameter settings are, regardless of their accuracies.

Concerning the actually obtained parameter values and the influence of the criteria chosen for tuning, we can state that the results on the CMA-ES as depicted in Figure 2 are obviously more consistent than the ones for TSC2 (Fig. 3), at least for the problems $F1$ and $F2$. It seems that tuning the CMA-ES on these 2 functions leads to very similar param-

Table 1: RMSE (lower is better), Pearson and Spearman correlation coefficients (closer to 1 are better) for the considered metaheuristics, functions and outputs.

Model	CMA-ES			TSC2		
	RMSE	Pearson	Spearman	RMSE	Pearson	Spearman
	$F1$, 2 global optima, 4 local ones					
	Number of peaks found					
SVM	0.33	0.87	0.87	0.65	0.74	0.76
Linear	0.31	0.88	0.89	0.93	0.34	0.22
Regression trees	0.38	0.83	0.87	0.55	0.82	0.69
Kriging	0.31	0.88	0.89	0.36	0.92	0.89
	Peak accuracy					
SVM	0.8	0.87	0.85	1.12	0.84	0.88
Linear	0.94	0.82	0.8	1.5	0.66	0.67
Regression trees	0.88	0.83	0.82	0.67	0.94	0.9
Kriging	0.94	0.82	0.8	0.65	0.94	0.92
	Distance accuracy					
SVM	0.67	0.87	0.85	1.08	0.75	0.82
Linear	0.81	0.81	0.79	1.46	0.51	0.55
Regression trees	0.74	0.84	0.79	0.55	0.94	0.88
Kriging	0.81	0.81	0.79	0.54	0.95	0.88
	$F2$, 1 global optimum, 9 local ones					
	Number of peaks found					
SVM	0.24	0.65	0.63	1.46	0.84	0.88
Linear	0.23	0.65	0.6	2.21	0.52	0.53
Regression trees	0.23	0.63	0.65	1.46	0.82	0.66
Kriging	0.23	0.65	0.6	0.72	0.96	0.95
	Peak accuracy					
SVM	5.41	0.71	0.73	4.14	0.7	0.84
Linear	5.02	0.74	0.79	5.09	0.47	0.54
Regression trees	5.24	0.74	0.77	2.85	0.9	0.8
Kriging	3.93	0.85	0.88	1.51	0.97	0.91
	Distance accuracy					
SVM	1.12	0.93	0.94	2.85	0.66	0.7
Linear	2.9	0.53	0.47	3.16	0.56	0.53
Regression trees	1	0.96	0.93	1.7	0.92	0.63
Kriging	0.74	0.97	0.97	0.5	0.99	0.77
	$F3$, 8 global optima, many local ones (only global considered)					
	Number of peaks found					
SVM	0.42	0.56	0.6	-	-	-
Linear	0.39	0.34	0.44	-	-	-
Regression trees	0.3	0.7	0.46	-	-	-
Kriging	0.39	0.34	0.44	-	-	-
	Best fitness					
SVM	-	-	-	118.04	0.9	0.91
Linear	-	-	-	167	0.81	0.83
Regression trees	-	-	-	163.23	0.82	0.79
Kriging	-	-	-	152.95	0.85	0.83
	Distance accuracy					
SVM	8.5	0.53	0.71	3.37	0.94	0.9
Linear	8.3	0.44	0.56	6.29	0.83	0.88
Regression trees	7.19	0.63	0.63	5.68	0.85	0.82
Kriging	4.86	0.86	0.8	2.3	0.98	0.95
	Overall					
SVM	4.46	5.99	**6.18**	4.03	6.37	6.69
Linear	5.43	5.21	5.34	8	4.7	4.75
Regression trees	3.72	6.16	5.92	3.46	7.01	6.17
Kriging	**2.75**	**6.18**	6.17	**0.71**	**7.56**	**7.1**

eter values, regardless of the chosen performance criterion and employed surrogate models. For F3, this is similar, but at least the parameter sets 'suggested by' the different models are a bit more similar for the distance accuracy criterion than for the average number of peaks. Interestingly, the actually obtained parameters differ a bit between $F1/F2$ and $F3$. For the former, large populations and increment factors are chosen with small step sizes, for F3 (with some exceptions, e.g. SVM model for the ap criterion), all parameters are set to medium to high values.

The results for TSC2 for $F1$ and $F2$ are overall better than those obtained by the CMA-ES as concerns all 3 criteria chosen for them. This is also reflected in a wide set of parameter settings (especially for $F1$, which has a smooth surface, perfect for TSC2 multimodality detection mechanisms) that are proper for finding good results, hence the apparent inconsistency of the results in the first two columns from Figure 3. For $F3$, the situation changes, the number of dimensions is high, TSC2 needs far more fitness evaluations to reach an optimum than CMA-ES does, so the GA-based tuning conducts, for most models, to very similar parameter sets, consequently similar results and resembling trends in column 3 of Figure 3.

5. CONCLUSIONS AND OUTLOOK

It is interesting to observe that for the parameter setting

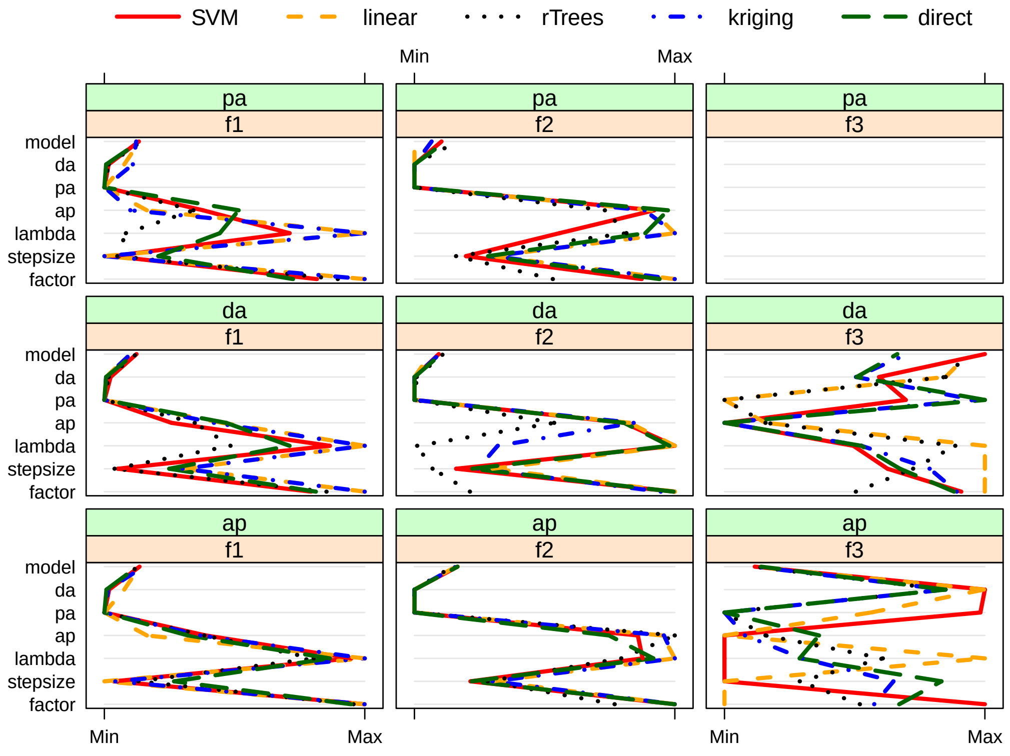

Figure 2: Parameter values and performances for different models, test problems, and performance criteria for CMA-ES (da = distance accuracy, ap = average number of peaks, pa = peak accuracy)

found by the GA, the validation does not usually go very far from the model prediction and, when it does, the differences are quite similar for most models, including direct tuning. Surely, there is a high amount of noise in the data, as the approaches can in some cases block into local optima, while other times they can be luckier. Further investigation regards: The enhancement of the accuracy of the models by including evolution control within the GA optimizer fitness and the replacement of the GA with a multicriterial optimization approach for searching for parameter sets that are appropriate for conflicting objectives (see e.g. the obvious difference in parameter sets for da and bf in $F3$, Figure 3).

Concerning model suitability for tuning the multimodal optimization performance, we can state that on rather simple functions (F1 and F2), the influence of the chosen model is not that large. SVM, regression trees and kriging all work sufficiently, with an advantage for kriging. The linear model is seemingly not very well suited, as expected. For the more difficult F3, the differences get even more emphasized. Considering the different performance criteria, the distance accuracy can be recommended as it works well for all situations. Overall, we can state that adjusting parameters for multimodal performance with such simple models indeed makes sense, which is good news for real-world applications (however, in this case one will have to think of new performance criteria as without knowledge of the optima, distance accuracy is not computable). A GA as tuning tool may not be the best choice, but is also sufficient in this case.

6. REFERENCES

[1] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation: The New Experimentalism*. Natural Computing Series. Springer, 2006.

[2] B. Bischl, O. Mersmann, H. Trautmann, and C. Weihs. Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evol. Comput.*, 20(2):249–275, June 2012.

[3] M. Breaban and H. Luchian. Pso under an adaptive scheme. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK*, pages 1212–1217. IEEE Press, 2005.

[4] W. Carpenter and J.-F. Barthelemy. A comparison of polynomial approximation and artificial neural nets as response surface. Technical Report 92-2247, AIAA, 1992.

[5] W. Carpenter and J.-F. Barthelemy. Common misconceptions about neural networks as approximators. *ASCE Journal of Computing in Civil Engineering*, 8(3):345–358, 1994.

[6] S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, Jan. 2001.

[7] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, , and A. Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2010.

[8] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[9] A. Fialho. *Adaptive Operator Selection for Optimization*. PhD thesis, Université Paris-Sud XI, Orsay, France, December 2010.

[10] A. Giunta and L. Watson. A comparison of approximation

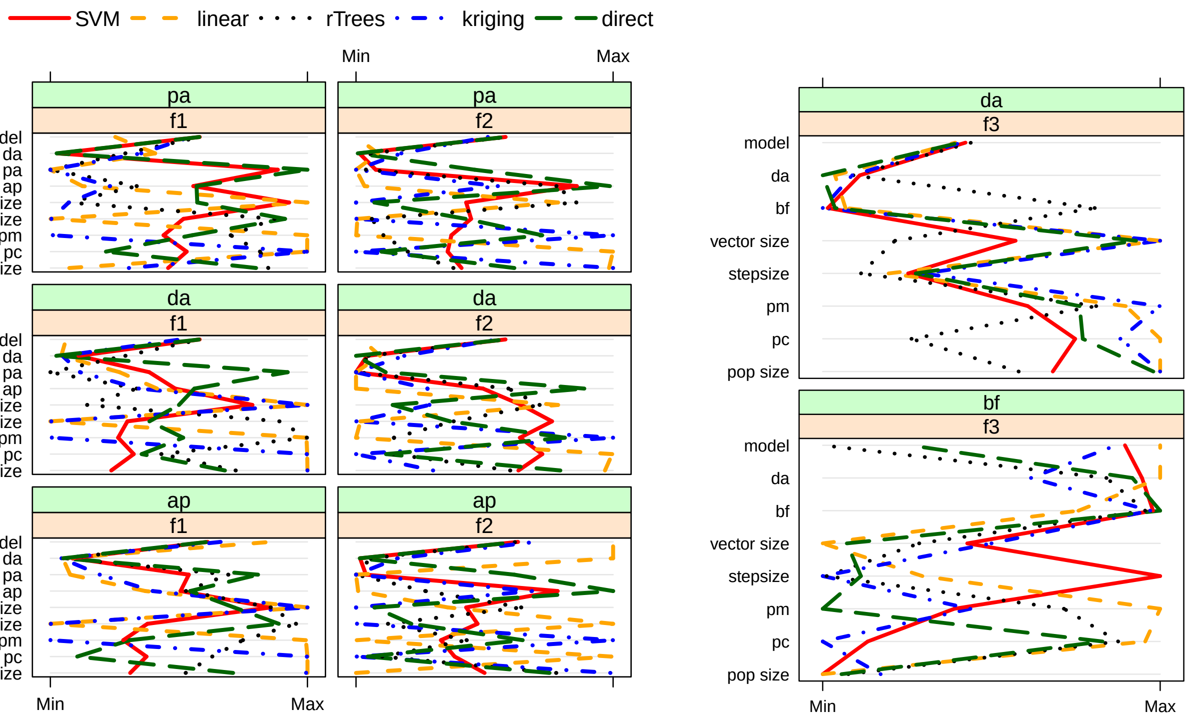

Figure 3: Parameter values and performance overview for TSC2 (bf is best fitness)

modeling techniques: Polynomial versus interpolating models. Technical Report 98-4758, AIAA, 1998.

[11] N. Hansen. The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[12] R. Jin, W. Chen, and T. Simpson. Comparative studies of metamodeling techniques under miltiple modeling criteria. Technical Report 2000-4801, AIAA, 2000.

[13] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9:3–12, 2005.

[14] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[15] Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00)*, pages 786–793, 2000.

[16] Y. Jin and B. Sendhoff. Reducing fitness evaluations using clustering techniques and neural network ensembles. In K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. K. Burke, P. J. Darwen, D. Dasgupta, D. Floreano, J. A. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, and A. M. Tyrrell, editors, *GECCO*, volume 3102 of *Lecture Notes in Computer Science*, pages 688–699. Springer, 2004.

[17] O. Kramer. Evolutionary self-adaptation: a survey of operators and strategy parameters. *Evolutionary Intelligence*, 3(2):51–65, 2010.

[18] C. Pintea and D. Dumitrescu. The importance of parameters in ant systems. *Int. J. Comput. Commun.*, 1(S):376–380, 2006.

[19] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evolutionary Computation*, 13(2):398–417, 2009.

[20] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *Parallel Problem Solving from Nature - PPSN V*, pages 87–96, 1998.

[21] O. Roustant, D. Ginsbourger, and Y. Deville. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51(1):1–55, 2012.

[22] W. Shyy, P. K. Tucker, and R. Vaidyanathan. Response surface and neural network techniques for rocket engine injector optimization. Technical Report 99-2455, AIAA, 1999.

[23] T. Simpson, T. Mauery, J. Korte, and F. Mistree. Comparison of response surface and Kriging models for multidisciplinary design optimization. Technical Report 98-4755, AIAA, 1998.

[24] C. Stoean, M. Preuss, R. Stoean, and D. Dumitrescu. Multimodal optimization by means of a topological species conservation algorithm. *IEEE Trans. Evolutionary Computation*, 14(6):842–864, 2010.

[25] T. M. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning*, 2011.

[26] S. Wessing, M. Preuss, and G. Rudolph. When parameter tuning actually is parameter control. In N. Krasnogor and P. L. Lanzi, editors, *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011*, pages 821–828. ACM, 2011.

[27] A. E. X. Li and M. Epitropakis. Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization. Technical report, RMIT University, Evolutionary Computation and Machine Learning Group, Australia, 2013.