Two-layered Surrogate Modeling for Tuning Optimization Metaheuristics

Günter Rudolph, Mike Preuss, Jan Quadflieg

I. Introduction

The problem of detecting suitable parameters for metaheuristic optimization algorithms is well known long since. As these nondeterministic methods, e.g. *evolution strategies* (ES) [1], are highly adaptible to a specific application, detecting good parameter settings is vital for their success. Performance differences of orders of magnitude (in time and/or quality) are often achieved by means of automated tuning methods. In the last years, several of these have been suggested, and many incorporate surrogate models for the algorithm parameter space. Although tuning methods are reliable tools to build specific optimization algorithms by modifying the parameters of canonic ones, their use is somewhat restricted to relatively cheap objective functions (in terms of computational cost). As a huge number of algorithm runs is necessary, they are simply not applicable when the evaluation time of the objective function increases beyond a certain level. If the objective function is too expensive for applying tuning methods directly, one can resort to simpler approaches:

1) Try to find a suitable parameter setting for a canonical metaheuristic by means of a simple randomized or space-filling design, or
2) give up on the parameter optimization approach and optimize the problem directly with a heuristic or metaheuristic that incorporates surrogate models itself.

The latter methodology is employed e.g. by EGO [2], but also by surrogate model enhanced metaheuristics like the *model-assisted evolution strategy* (MAES) [3]. Whereas tuning methods utilize many runs of standard algorithms which are performed on the original optimization problem and do not model it[1], MAES and similar algorithms model the optimization problem 'on-the-fly' but do not adapt the algorithm itself. There is no reason to believe that these two approaches could not go well together and thus make tuning applicable to more expensive optimization problems than before. In this work, we combine the use of metamodels (= surrogate models) for the tackled problem with the tuning technique SPO [4] which uses kriging metamodels in the algorithm parameter space. Note that this combination is especially straightforward for a model-assisted optimization algorithm, but that the technique is useful also if other optimization algorithms shall be applied to a costly problem and need to be tuned.

All authors are with the Chair of Algorithm Engineering, Computational Intelligence Group, Dept. of Computer Science, Technische Universität Dortmund, Germany. E-mail: firstname.lastname@tu-dortmund.de

II. Two-layered model-supported optimization

The basic idea of our approach is to start an algorithm tuning process with a minimal space-filling design and to use the obtained objective function samples to build a first-layer surrogate (kriging) model on which the algorithm tuning is then continued. Modern tuning algorithms like SPO themselves establish a surrogate model we term second-layer model here, as it does not operate on the problem itself. As soon as the tuning process converges, validation runs are performed to test the tuned algorithm on the real objective function. Thereby, more samples become available for updating the first-layer model. The tuning process may then be continued on the updated model and the resulting algorithm configuration validated again. This loop shall be terminated if either a predefined budget of objective function evaluations is used up or no better algorithm configurations are obtained from the tuning process any more.

[1] their model belongs to the algorithm parameter space

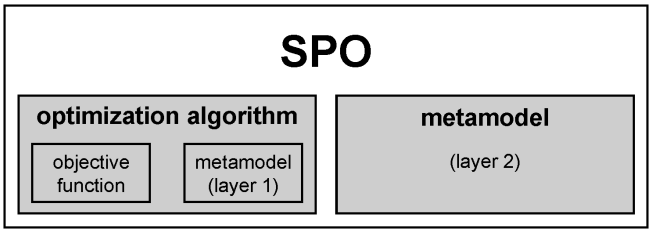

Fig. 1. Two layers of metamodels for saving function evaluations.

We make the assumptions that the algorithm tuning problem is easier to solve than the optimization problem itself and that a surrogate model represents the real problem good enough to allow for tuning the optimization algorithm to perform well on the original problem. The first assumption is supported by the smaller dimensionality of the tuning problem (usually around 5, many real-world optimization problems have between 10 and 30 variables), and prior knowledge about the mechanisms of the optimization algorithms. The second assumption has to be tested experimentally. It is clear that although we head for greatly reduced tuning times, our approach is suitable only if objective function evaluations are on the order of minutes at most; otherwise, tuning is not possible any more and few runs of any optimization algorithm have to suffice.

In this preliminary study, we restrict ourselves to validate the second assumption. It is currently unknown how detailed a first-order model has to be to allow meaningful tuning. A very accurate model will surely suffice, but it is also very expensive. The overall question is if the number of objective function evaluations to invest into the model is small enough. At most, it should be on the order of 10-20 algorithm runs, otherwise using the tuning method directly on the problem is cheaper, if applicable at all.

We can expect that the necessary accuracy degree of the first-order model will not be the same for all problems, and that some problems may possess properties rendering the whole approach infeasible. Furthermore, not every surrogate model may be suitable for a certain problem type. Thus we start with two well-known benchmark problems, the rotated Rastrigin problem and Schwefels problem 2.13, both employed in the CEC'05 benchmark suite for evolutionary optimization algorithms [5]. Both are multimodal and the first has the overall structure of a hyper-parabola with many bumps, whereas the second is less multimodal but deceptive (the best optima are on different sides of the search space). For these problems, generating point databases for setting up surrogate models takes only minutes. In a second experiment, the approach is tested on a real-world problem where a standard SPO tuning procedure with e.g. 500 algorithm runs would not be applicable because each run takes around a day. The task is to find a suitable algorithm for solving problems from fluid dynamics, in our case optimizing a ship propulsion system.

For both levels of surrogate models, we rely on ordinary kriging models (assuming an unknown constant trend). However, the use of kriging as first-order surrogate model is rather motivated by the success of the method as second-order model within SPO than by factual evidence. Nevertheless, kriging may be a good candidate as it makes very few assumptions on the modeled function and thus should work reasonably well in many cases.

III. Model usage in MAES and SPO

In order to clarify how the different algorithms use surrogate models, we shall specify how they work in detail. The MAES is a model-enhanced evolution strategy that learns the step sizes in a self-adaptive fashion as described in [1]. It employs a population of μ solutions from which λ new search points are generated by gaussian mutations (with standard deviation given by the current step size, initially set to σ_{init}) in all dimensions simultaneously. Before applying mutation, the step size undergoes a slight randomized change regulated by the learning rate τ. The μ most successful offspring (new search points) are kept and form the next population. The parameter κ denotes the maximum number of iterations any solution may survive in the population. See table II, row of entry 'standard' for the default settings used in this work.

During runtime, the MAES keeps a database of points ever visited and uses it to establish a first order surrogate model of the problem it operates on. We rely on a local ordinary kriging model made from the 15 points in the database with the smallest euclidean distance to the new search point

in this work, but other models would be possible as well, given that a surrogate model component is added. With a given rate (here: 0.5), the MAES samples the original objective function, otherwise it does so on the kriging model. One advantage of using a kriging model lies in its ability to return an estimate of the error next to one of the function value. The estimated function value and the error estimation are added so that the MAES prefers regions with predicted good function values, but also honors large errors that may result in good function values if sampled on the real objective function. For the case of minimization, the error value is subtracted from the function value estimation. In any case, the importance of the error estimation is scaled by a constant we keep at 1 in this work, as advocated by preliminary experiments. Note that the internal model kept by the MAES inspired sampling on the kriging model only as we do later on. However, every other derivative-free optimization algorithm may be used as well. When sampling on the model only, we allow for twice as many function evaluations as the MAES was given in the mixed case, corresponding to the rate of 0.5 real and 0.5 model-based evaluations.

The employed tuning method SPO starts with a latin hypercube design in the provided algorithm parameter space and then establishes a global ordinary kriging model to predict good regions for further sampling. It utilizes the expected improvement criterion of EGO [2] to take the predicted error value into account as well as the function value prediction. A huge difference between MAES and SPO in addition to their different working spaces (problem vs. algorithm parameters) is that SPO has to cope with non-deterministic answers, whether MAES expects deterministic ones. It does so by doing increasing numbers of repeats at good detected algorithm configurations (parameter values). Thus, the most successful configurations are sampled many times more than had been undertaken for the initial design. The obtained best function values for each performed algorithm run are added to the model so that it improves over time.

SPO also possesses some parameters that are set as follows in this work: The total budget of algorithm runs is 500, the initial design has a size of 25 with 4 repeats each. In each iteration, only one new point is added and is run two times as often as the current leader (which is also re-run up to the same amount of runs to attain a fair comparison), with a maximum of 30 runs per configuration.

IV. Experiment on Benchmark Problems

For two well-known benchmark problems (each in 2 and 10 dimensions), we first generated a number of points to be used for modeling via performing repeated runs. To prevent too many very similar points entering the databases, a minimum Manhattan distance was required in at least one dimension or the fitness value to every existing point (relative to the extension of the whole population). The minimal distance was set to 0.0025 for 2 dimensions, and 0.01 for 10 dimensions. For every problem, databases of sizes 0, 1000 and 2000 were established.

Research Question. Is tuning on first-order models of the problem successful with respect to algorithm behavior on the original problem?

Pre-Experimental Planning. At first, we tested if providing the MAES with a database of points for its internal surrogate model improves the optimization behavior. The MAES used kriging model samples and real evaluations each with probability 0.5. The parametrization is found in table II as *standard* and is the result of manual tuning on the different problems, including the real-world problem reported later on.

Figure 2 shows box plots of the results of 20 runs for each configuration with all three database sizes (configuration 1,2,3 means DB size 0, 1000, 2000). Although an improvement seems visible for a large database in 10d of f10, the result sets showed no significant difference by means of a Wilcoxon rank-sum test at significance level 0.05. For 10d on f12, significance was also not reached. In 2d for f12, configuration 3 was significantly better than any other, and for f10, configuration 2 was better than any other. This is somewhat unexpected but may pay tribute to the highly multimodal nature of the test functions. One may deduct that except for few cases, adding databases does not change the performance of the MAES much on difficult functions. However, the MAES parameter setting may have been not ideal for

that case so that databases get more important for specific algorithm configurations.

Task. Our hypothesis is that the behavior of the MAES on the benchmark problems can be improved via tuning it on a first-order surrogate model. We require that a significant improvement results in precision qualities at the 5%-level, according to a Wilcoxon rank-sum test.

Setup. All runs on 10 dimensional problems are terminated after 300 function evaluations on the original problem, and 600 on the surrogate model, in 2 dimensions, these limits are set to 60 and 120, respectively. The higher values for the runs on the model are fair as the MAES on the original problem is configured to do half of its evaluations on its internal model. The overall very low function evaluation numbers stem from the need to apply the method to costly real-world problems later on, which often do not allow for more than a few hundred evaluations. For 2 dimensions, the budget has been reduced deliberately to increase the hardness of the problems, as $300/600$ evaluations often lead to near optimal optimimization results where tuning is not needed any more.

On each of the 4 test problem settings (f10 and f12 in 2 and 10 dimensions), three SPO tuning processes are performed with a budget of 500 algorithm runs, separately for three configurations. These are: MAES without initial database (as is normally the case) on the original problem, MAES on a kriging first-order surrogate model of the problem with a database size of 1000, and the same again with a database size of 2000. The second model should be more accurate, whether the tuning on the original problem may serve as the uppper limit of the performance to be reached when tuning on the model only. After tuning, the best resulting configurations are validated with 20 runs on the original problem. The tuning intervals are given in table I. They allow for a wide range of MAES variants, only the population size/offspring number is fairly restricted as the number of function evaluations allowed is very small and the MAES should be able to run for at least 5-10 generations.

Experimentation/Visualization. Figure 3 displays box plots of the validation runs after SPO tuning in a similar fashion as for the original MAES runs in figure 2. Table II shows the parameter values chosen by SPO for the most successful tuning runs (as indicated by figure 3. Table III reports the p-values of Wicoxon rank-sum tests between the standard configuration and the SPO determined configurations.

Observations. Four all 4 tested function/dimension combinations, tuning results in significantly better configurations. Pondering the determined MAES parameter values as reported in table II leads to the insight that generally a larger population (and thereby enabled recombination) is helpful. Interestingly, the parameter sets found while tuning only on the surrogate model are relatively close to the ones determined on the original function. The success of the tuning process if run on the model seems to depend on the number of points put into it, and not always, more points are better (f10-10d). However, this is always the case for the problem f12. It seems that the advantage gained here is mainly due to a more stable behavior (smaller spread). At least one of the model-tuned configurations was significantly better than the original parameter set, except for f10-2d, where significance was narrowly missed with a p-value of 0.059.

Discussion. The overall impression is that the two-layered approach for tuning works remarkably well. Although the used first-order surrogate models with 1000 and 2000 points may not be very accuracte, given that the problems are highly multimodal, an improvement is reached in 3 of four cases (probably mainly by switching from a single-individual population to a larger one). On f10-2d tuning may be very hard because the standard parameters already entail good results, even if there is still room for a little improvement. The observation that for f12, more accurate models provide better tuning results meets expectation well. However, this seems to be different for f10. As possible explanation, it comes to mind that SPO is itself

TABLE I

PARAMETER INTERVALS FOR THE TUNING PROCESS, NAMELY INITIAL STEPSIZE, LEARNING RATE, MAXIMUM ALLOWED LIFETIME, POPULATION SIZE, OFFSPRING NUMBER.

σ_{init}	τ	κ	μ	λ
0.01:0.5	0.1:2	1:30	1:5	5:15

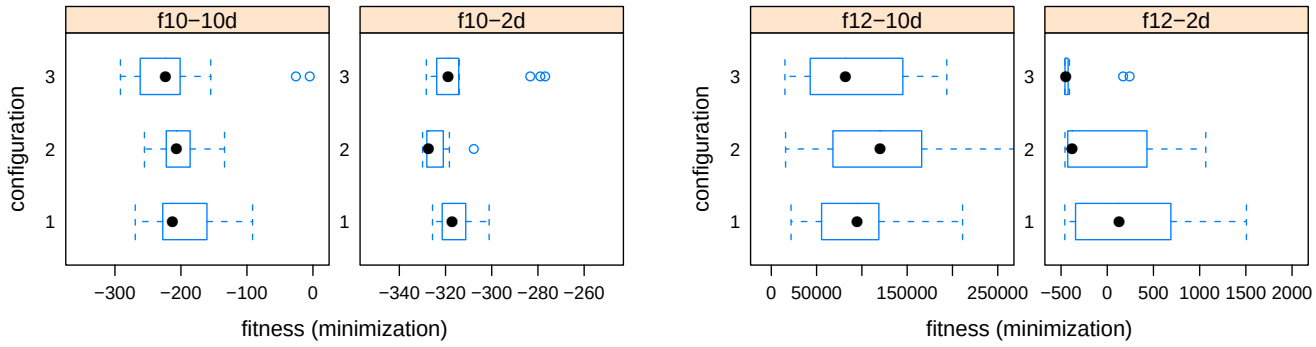

Fig. 2. Performance of the MAES with different additional databases, configuration 1,2,3 stands for 0, 1000 and 2000 points. Left: Problem f10 (Rastrigin, optimum -330), 2 and 10 dimensions, right: Problem f12 (Schwefel, optimum -460), 2 and 10 dimensions.

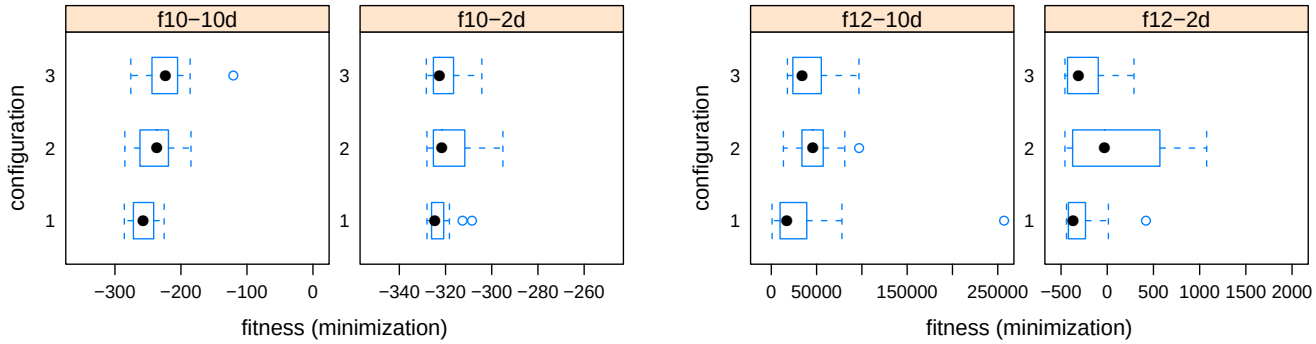

Fig. 3. Performance (20 validation runs) of the SPO-tuned MAES configurations, 1 stands for SPO on the original problem, 2 for SPO on the 1000 point database model, and 3 for SPO on the 2000 point database model. The boxes shall be compared with configuration 1 of figure 2 that shows the performance of the untuned algorithm. Left: Problem f10 (Rastrigin, optimum -330), 2 and 10 dimensions, right: Problem f12 (Schwefel, optimum -460), 2 and 10 dimensions.

a non-deterministic method, so that some tuning runs are simply luckier than others. Nevertheless, the model size is a question that deserves further attention in future.

V. A Real-World Test Problem

As a suitable test problem, we employ the optimization of a relatively new ship propulsion system (a linearjet, figure 4) which possesses 15 design variables. It consists of a tube with a rotor and a stator, and several lengths, angles and thicknesses can be variated. Our objective function is a very basic fluid dynamic simulation of a linearjet that takes about 3 minutes to compute, and the task is to reduce cavitation at a predefined efficiency. Cavitation (the emergence of vacuum bubbles in case of extreme pressure differences due to very high accelerations of the water) damages the propulsion system and also leads to high noise levels. Note that for bringing down the simulation time to minutes, many simplifications are in effect, so that the obtained result is not very accurate. A full simulation would take about 8 hours, employing parallel computation, which is by far too much to serve as test problem. However, the simplified simulation is still accurate enough to detect good design points that can afterwards validated by the full simulation.

A MAES [3] has been applied to a simpler form (less variables) of the problem with limited success [6]. This may be due to a very rugged search space with many plateaus, cliffs and bumps. As indicated by a random sample around a good search point in figure 5, the quality to distance correlation looks fairly unstructured and layered. This impression is approved by several grid tests

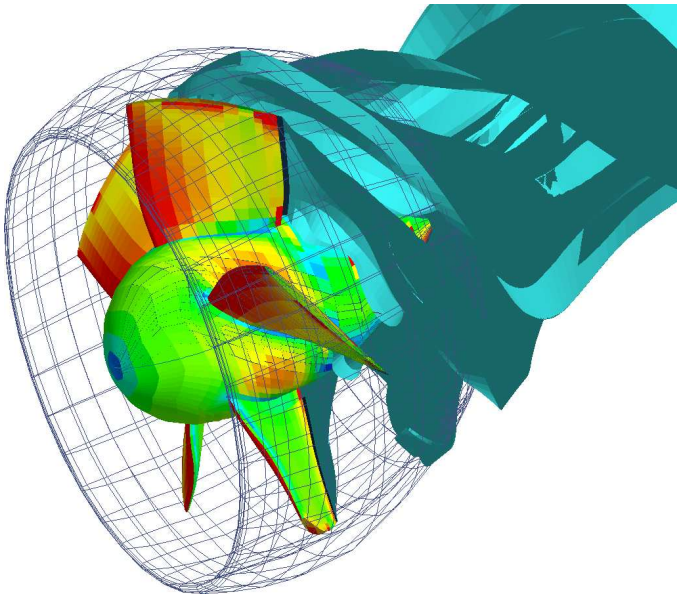

Fig. 4. Vizualization of the running linearjet propulsion system simulation. Due to symmetries, only a partial movement is simulated, and the water body is restricted to the parts in touch with the tube, rotor and stator.

in only 2 dimensions around a good solution as displayed in figures 6 and 7. The two closely correlated variables of the rotor printed in figure 6 reveal a very rugged surface. Small steps into either direction may lead to high cavitation which is heavily penalized, with no recognizable smooth transition between them. The other figure leads to a very different impression with linear structures of bad fitness. Overall, we deduct that the fitness surface is fairly inhomogenous, which should make it difficult for optimization algorithms as well as for providing good models of it. However, if used

TABLE II

STANDARD AND TUNED PARAMETER SETTINGS OF THE MAES (INITIAL STEPSIZE, LEARNING RATE, MAXIMUM ALLOWED LIFETIME, POPULATION SIZE, OFFSPRING NUMBER).

	σ_{init}	τ	κ	μ	λ
standard	0.15	1.0	10	1	5
f10 2d SPO(O)	0.088	0.753	6	3	13
f10 2d SPO(1000)	0.338	1.662	15	3	14
f10 10d SPO(O)	0.400	0.525	19	5	11
f10 10d SPO(1000)	0.415	1.525	20	5	12
f12 2d SPO(O)	0.055	0.114	20	5	14
f12 2d SPO(2000)	0.3974	0.696	29	4	14
f12 10d SPO(O)	0.017	0.173	23	5	13
f12 10d SPO(1000)	0.208	0.210	10	4	11
linearjet SPO(2000)	0.170	0.794	20	4	11

TABLE III

P-VALUES OF THE WILCOXON RANK-SUM TEST BETWEEN THE 20 VALIDATION RUNS OF THE DIFFERENT PARAMETER SETS, STANDARD AND DETECTED BY SPO. SPO(O) MEANS TUNING ON THE ORIGINAL PROBLEM, SPO(1000) TUNING ON 1000 POINT, SPO(2000) ON 2000 POINT SURROGATE MODEL.

	f10 2d	f10 10d	f12 2d	f12 10d
standard : SPO(O)	10^{-3}	10^{-5}	0.010	10^{-5}
standard : SPO(1000)	0.134	0.005	0.799	10^{-4}
standard : SPO(2000)	0.059	0.277	0.013	10^{-4}
SPO(O) : SPO(1000)	0.142	0.024	0.021	0.006
SPO(O) : SPO(2000)	0.883	10^{-4}	0.038	0.013
SPO(1000):SPO(2000)	0.142	0.102	0.883	0.369

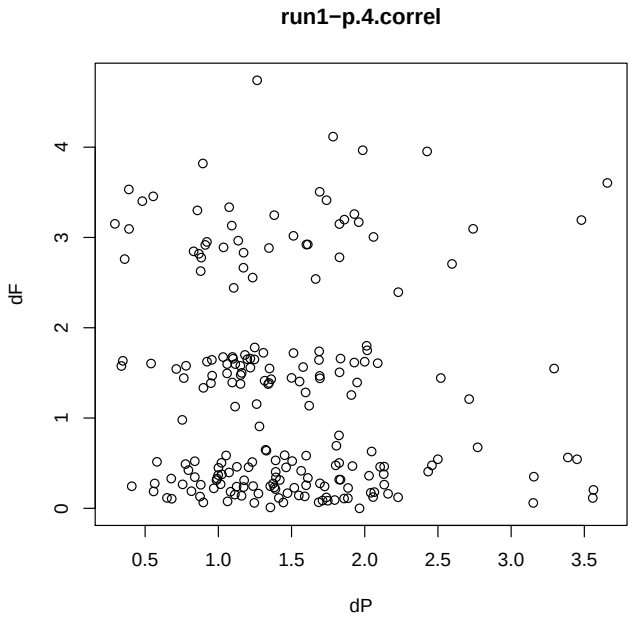

Fig. 5. Quality to search space distance correlation around a good solution, Δquality (dF) over Δsearch space distance (dP).

in a tuning context, a surrogate model only has to embody the most important properties of the original problem. The tuning process may then adapt the algorithm to these if parameter changes provide this possibility. Without looking at the following experiment, we cannot be completely sure if this is the case at all for the system of MAES and the linearjet optimization problem.

VI. VALIDATION EXPERIMENT ON THE LINEARJET PROBLEM

The original goal of this work had been to find a good parameter setting and thus a good

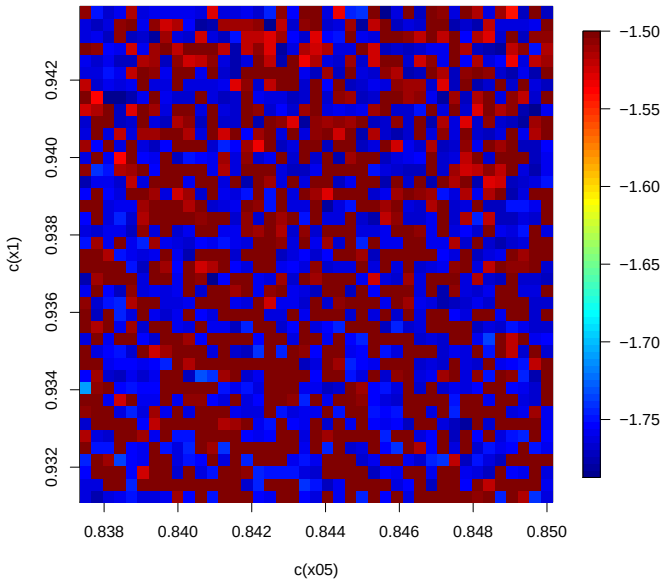

Fig. 6. 20x20 grid test around a very good point in dimensions 11 and 12 (2 rotor parameters), minimization.

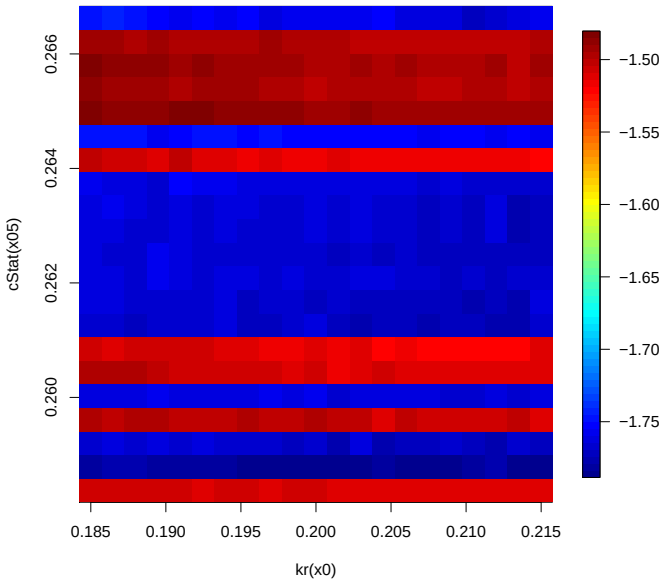

Fig. 7. 10x10 grid test around the same point as in figure 6, dimensions 13 and 20 (which should be rather uncorrelated), minimization.

algorithm for optimizing on the linearjet problem. After showing that our 2-level surrogate model approach works on simple benchmark functions, the test on the original real-world problem is the necessary next step. However, we cannot conclude that the approach is either always successful or completely useless, regardless of the outcome.

Research Question. Can the 2-level surrogate model based tuning approach also improve algorithms on real-world problems?

Pre-Experimental Planning. The standard parameter setting of the MAES used throughout the whole paper was determined via first runs on this problem.

Task. We demand that the algorithm with tuned parameters performs significantly better in terms of best fitness per run than the untuned version, at the 5 % level of a Wilcoxon rank-sum test.

Setup. A database of 2000 points is generated from previous test runs on the linearjet problem, from which we create a local ordinary kriging model. The SPO tuning operates on this model only, with a run length of 600 evaluations (according to the 0.5 probability for the MAES in its chosen configuration to sample from the real objective function and a run length of 300 real function evaluations). The MAES is then run 9 times each with the obtained best configuration and the standard configuration for validation purposes with run length 300. Note that one run needs between 12 and 24 hours computation time on a modern PC.

Experimentation/Visualization. The best parameters resulting from tuning on the model only are given in table II, row 'linearjet SPO(2000). Figure 8 shows a box plot comparing the distributions of the 9 validation runs each. The mean values of the best solutions per run are -1.4812 vs -1.5342, the standard deviations 0.0539 vs 0.0478. The two samples are different with a p-value of 0.02 in a Wilcoxon rank-sum test (the data is not normal as determined by a Shapiro-Wilk test with a p-value of 10^{-3} on the standard parameter results).

Observations. We find that the parameters obtained from tuning the MAES on the surrogate model of the linearjet problem are remarkably similar to the ones found on the benchmark functions, except a more moderate $\sigma_{init} = 0.17$, which corresponds well to the original value of 0.15. The validation runs show a different type of distribution for the standard and the tuned parameter setting. Under standard parameters, most values

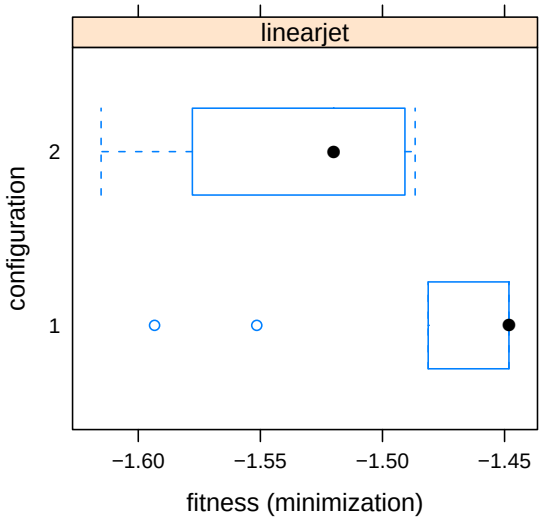

Fig. 8. Performance of the MAES on the linearjet problem before (configuration 1) and after tuning (configuration 2). The dots correspond to the median values of the distributions.

are concentrated around the median, with some outliers. However, the spectrum of values in the tuned results is much broader.

Discussion. The validation experiment is successful, as the SPO-tuned parameter values lead to a significant performance increase. The different distributions of the validation runs probably stem from the type of algorithm chosen. The tuned MAES resembles a model-enhanced (4,11)-ES with $\kappa = 20$ and so may profit from a more globally oriented behavior than the standard parameter setting with a population size of one. It seems to pay off to cling longer to the best found search points (κ 20 vs 10). However, the observed success does not mean that there is no other evolutionary optimization method that would perform even better. We have only obtained a good configuration for the MAES. Nevertheless, even that was not fully expected, as the objective function does not look very well suited for any model-based optimization scheme.

VII. Conclusions

The results from our experiments on some benchmark functions and a difficult real-world problem support our conjecture that the concept of using a two-layered surrogate model approach for tuning optimization algorithms possesses a fruitful and as yet unexplored potential for optimization under scarce resources. It remains to provide evidence that parameter tuning via a surrogate model works since only the main characteristics of the true problem must be reflected by the surrogate model. If so, every objective function evaluation saved by using a surrogate model in the optimization runs can be invested for tuning the algorithm. As a consequence, using the same number of function evaluations much better solutions can be found than without the two-layered approach.

As future work, we envision establishing an integrated method for using a first-order surrogate model within the tuning process from the different components used separately within this first study (first runs on the original problem, tuning process, validation runs). This would enable tuning on costly objective functions in a systematic manner, which is not possible currently.

Acknowledgments

Mike Preuss gratefully acknowledges support from the Deutsche Forschungsgemeinschaft (DFG), grant no. RU 1395/3-2 “Ein Verfahren zur Optimierung von aus Mehrkomponenten bestehenden Schiffsantrieben”.

References

[1] H.-G. Beyer and H.-P. Schwefel, “Evolution strategies: A comprehensive introduction,” *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[2] D. Jones, M. Schonlau, and W. Welch, “Efficient global optimization of expensive black-box functions,” *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[3] M. Emmerich, K. Giannakoglou, and B. Naujoks, “Single- and multi-objective evolutionary optimization assisted by gaussian random field metamodels,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 421–439, 2006.

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

[5] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, “Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization,” Nanyang Technological University, Singapore, Tech. Rep., May 2005, http://www.ntu.edu.sg/home/EPNSugan.

[6] B. Naujoks, M. Steden, S.-B. Müller, and J. Hundemer, “Evolutionary optimization of ship propulsion systems,” in *Proc. 2007 Congress on Evolutionary Computation (CEC’07), Singapore*. Piscataway NJ: IEEE Press, 2007.