Evolution Strategies with Controlled Model Assistance

H. Ulmer, F. Streichert, A. Zell
Centre for Bioinformatics Tübingen (ZBIT)
University of Tübingen
Sand 1, 72076 Tübingen, Germany
Email: ulmerh@informatik.uni-tuebingen.de

***Abstract*— Evolutionary Algorithms (EA) are excellent optimization tools for complex high-dimensional multimodal problems. However, they require a very large number of problem function evaluations. In many engineering and design optimization problems a single fitness evaluation is very expensive or time consuming. Therefore, standard evolutionary computation methods are not practical for such applications. Applying models as a surrogate of the true fitness function is a quite popular approach to handle this restriction. It is straightforward that the success of this approach depends highly on the quality of the approximation model. We propose a Controlled Model Assisted Evolution Strategy (C-MAES), which uses a Support Vector Regression (SVR) approximation by pre-selecting the most promising individuals. The model assistance on the evolutionary optimization process is dynamically controlled by a model quality based on the number of correctly pre-selected individuals. Numerical results from extensive simulations on high dimensional test functions including noisy functions and noisy functions with changing noise level are presented. The proposed C-MAES algorithm with controlled model assistance has a much better convergence rate and achieves better results than the model assisted algorithms without model control.**

I. Introduction

Evolution Strategies (ES) are one class of Evolutionary Algorithms (EAs), which are often used as optimization tools for complex high-dimensional, multimodal problems [13] [14]. In contrast to other EAs like Genetic Algorithms or Genetic Programming ES work directly on real valued objective variables, which represent a possible solution. Therefore ES are very suitable for many engineering and design optimization problems.

However, like other population based EAs ES require a very high number of fitness function evaluations to determine an acceptable solution. In most-real world engineering optimization applications the process of fitness evaluation is very expensive and time consuming. Therefore standard ES methods are not practical for such applications.

A promising approach to make evolutionary optimization methods more practical is the application of modeling techniques, where a model evaluation is orders of magnitude cheaper than a true fitness function evaluation. A model is trained on already evaluated fitness cases and is used to guide the search for promising solutions. This approach decreases the number of expensive fitness evaluations and has a better convergence rate. The application of modeling techniques in evolutionary computation receives increasing attention [9] [3] [4] [16] [17]. The results achieved are very encouraging and show that model assistance enhances the performance of standard EAs.

The selection of an appropriate model to approximate the fitness function is very important. Only a good approximation model of the true fitness landscape can support the optimization process. A second important point is the coupling of the approximation model with the evolutionary optimization process, which manages how the optimization process is affected by replacing the expensive true fitness evaluation with the prediction of the model.

Our work aims at improving the model management by dynamically controlling the impact of the model on the evolutionary process. The control is based on a quality measure of the approximation model, which takes the ability of the model to identify the most promising individuals into account.

The remainder of this paper is organized as follows:

Fitness approximation by Support Vector Regression (SVR) is introduced in section II. Section III describes the basic Model Assisted Evolution Strategy framework. We empirically identify the problem of model impact control and introduce a framework, which dynamically controls the impact of the model assistance on the evolutionary optimization process through the size of the pre-selected population in section IV. Numerical results from extensive simulations on several high dimensional test functions including are presented in section V. The paper closes with a brief conclusion and outlook on future work.

II. Fitness Aproximation with Support Vector Regression (SVR)

Consider a d-dimensional real valued problem with a fitness function, which is to be minimized, and a given data set $\mathcal{D}$ of N already evaluated fitness cases $(\vec{x}_n, t_n)$. We want to predict the fitness $f(\vec{x})$ at a new unseen data point $\vec{x} \notin \mathcal{D}$.

Due to the limited amount of data and the high dimensionality of most problems in object space (curse of dimensionality), it is very difficult to obtain a perfect global approximation of the true fitness function. In such cases it is better to restrict to local models, which are only valid for a distinct area in object space.

It is obvious, that the better the model approximates the true fitness landscape, the better it supports the evolutionary optimization process. Moreover a bad model can mislead the optimization process.

Therefore, a suitable fitness approximation model must be carefully chosen. The model should fulfill several requirements [12]. Compared to the real fitness function it should have a small computational complexity and it should represent the global trends of the fitness landscape.

Various modeling techniques have been used for fitness approximation in evolutionary computation. Neural networks [9], [6], [7] including radial basis functions [17], [19] are widely used for fitness approximation in evolutionary optimization. Ong et al. [10] combines radial basis functions with transductive inference to generate local surrogate models. Gaussian Processing [3], [16] and Kriging [4], [12] are statistical modeling techniques, which are also used for fitness function approximation. A comparison of neural networks and kriging for fitness approximation in evolutionary optimization can be found in [18].

In the following we give a brief description of Support Vector Machines (SVM) [2], which are used in our study. The SVM regression algorithm seeks to estimate a linear function,

$$f(\vec{x}) = \langle \vec{\omega}, \vec{x} \rangle + b \tag{1}$$

based on the given training data. This is done by minimization of a regularized risk functional:

$$R[f] = \frac{1}{2} \|\vec{\omega}\|^2 + C \cdot R^{\epsilon}_{emp}[f] \tag{2}$$

$$R^{\epsilon}_{emp}[f] = \frac{1}{N} \sum_{i=1}^{N} max\{0, |t_i - f(\vec{x}_i)| - \epsilon\} \tag{3}$$

$R^{\epsilon}_{emp}[f]$ measures the ϵ-insentitive training error. C is a regularization constant determing the trade-off with the complexity penalizer $\|\vec{\omega}\|^2$. A small $\|\vec{\omega}\|^2$ corresponds to a flat function.

The minimization of the regularized risk functional (equation 2) is equivalent to a constraint optimization problem, which can be formalized by a lagrangian formalism and leads to a quadratic programming problem. A detailed description is given in [15]. The SVR output (equation 4) for the linear regression case is expressed in terms of the scalar product and a set of lagrangian multipliers $\alpha_i^{(*)}$.

$$f(\vec{x}) = \langle \vec{\omega}, \vec{x} \rangle + b = \sum_{i=1}^{N} (\alpha_i - \alpha_i^{\star}) \cdot \langle \vec{x}_i, \vec{x} \rangle + b \tag{4}$$

Nonlinear regression can be performed by introducing a kernel, which substitutes the scalar product and is often given as a gaussian kernel (equation 5).

$$k(\vec{x_i}, \vec{x}) = \exp\left(-\frac{\|\vec{x_i} - \vec{x}\|^2}{2\sigma^2} \right) \tag{5}$$

To guarantee a unique optimal solution to the quadratic optimization problem the kernel matrix $K = k(\vec{x}_i, \vec{x}_j)_{i,j=1,..,N}$ must be positive definite. The parameter σ of the Gaussian kernel, the value for ϵ and the regularization constant C has to be selected by the user or by a model selection procedure. Major advantages of the SVM over other methods are, that there are no local minima during training and that the generalization error does not depend on the dimension of the object space. In our study we used a SVR implementation called LIBSVM [1] written in Java.

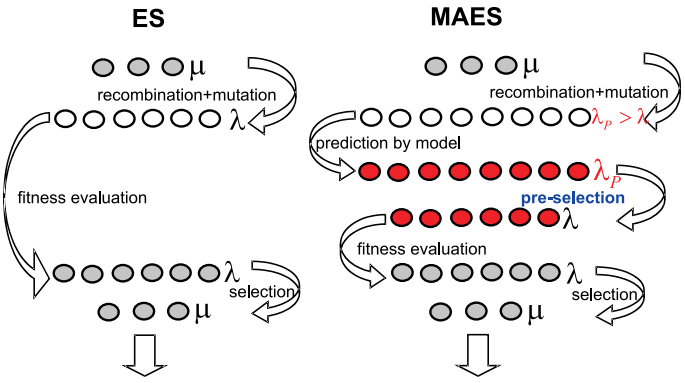

Fig. 1. Standard (μ, λ)-ES on left side and the model assisted extended (μ, λ)-MAES on the right side.

III. Model Assisted Evolution Strategies (MAES)

In the following we describe the Model Assisted Evolution Strategy (MAES) proposed by Ulmer et al.. A more detailed description of the algorithm is given in [16]. We start our consideration with a standard (μ, λ)-ES, which will be later extended to the model assisted ES.

A standard ES works on a population of potential solutions $\vec{x}$ (individuals) by manipulating these individuals with the evolutionary operators reproduction, recombination and mutation. λ offspring individuals are generated from μ parents. After evaluating the true fitness of the λ offspring individuals, a (μ, λ) strategy selects the best μ individuals to build the parent

Algorithm 1 Model Assisted Evolution Strategy (MAES) .

```
(1)  Procedure MAES
(2)  begin
(3)    eval=0;
(4)    Pop=CreateInitialPop();
(5)    Pop.EvalTrueFitness();
(6)    Model.update(Pop);
(7)    while(eval < maxeval)
(8)      PreOffspring=Pop.Reproduce(λ_Pre);
(9)      PreOffspring.Mutate();
(10)     PreOffspring.PredictFitness(Model);
(11)     RealOffspring=PreOffspring.SelectBest(λ);
(12)     RealOffspring.EvalTrueFitness();
(13)     Model.update(RealOffspring);
(14)     Pop=RealOffspring.SelectBest(μ);
(15)     eval=eval+λ;
(16)   end while
(17)  End
```

population for the next generation. The algorithm terminates, when a maximum number of fitness function evaluations have been performed.

The MAES is based on this standard (μ, λ)-ES and can be easily developed by introducing an additional pre-selection step based on the fitness predictions of the approximation model.

To incorporate the approximation model into the ES a pre-selection concept is used. Compared to the standard ES not λ but $\lambda_{Pre} > \lambda$ new offspring individuals are created from μ parents by applying the evolutionary operators reproduction, recombination and mutation (see algorithm 1 line 8). A graphical presentation of the algorithm compared to a standard (μ, λ)-ES is given in figure 1.

These λ_{Pre} individuals have to be pre-selected by the approximation model to generate the offspring of λ individuals (algorithm 1 line 10 and 11), which are then evaluated with the true fitness function (line 12).

The basic idea behind this approach is that only the most promising individuals with a good fitness prediction are evaluated with the true fitness function, which results in a reduction of the number of expensive true fitness calls.

The model is trained at the beginning with a randomly created initial population (line 6) and is updated after each generation step with λ new fitness cases (line 13). The algorithm terminates when a maximum number of fitness function evaluations have been performed.

For $\lambda_{Pre} = \lambda$, the algorithm performs like a standard (μ, λ) ES and the model has no impact on the ES. Increasing λ_{Pre} results in a larger selection pressure in the pre-selection and in a stronger impact of the model on the convergence behavior of the optimization process.

To analyze the influence of the pre-selection population size λ_{Pre} on the optimization process numerical simulations were performed for different values of $\lambda_{Pre} = 100, 200, 300, 400$.

All presented results are the mean of 100 repeated runs with different seed values for random number generation.

Throughout our study we used Main Vector Adaptation (MVA) [11] for step size control of the mutation operator without recombination and an initial population size of 10. MVA is a Covariance Matrix Adaptation [5] variant, which has the advantage of linear time and space complexity [11].

The approximation model was built in all cases by SVR with training data from the 3λ most recently performed fitness evaluations. For this reason the model is a local model of the individual's neighborhood in object space. Using more training data improves the performance only slightly, but results in much higher computational costs for model training.

In figure 2 you can see the results for the 10-dimensional sphere test function ($f_{sphere}(\vec{x}) = \sum_{i=1}^{10} x_i^2$). A very high dependency of the results on the value of parameter λ_{Pre} can be observed. The bigger λ_{Pre}, is the better the algorithm performs, because the sphere fitness landscape can be approximated very well by SVR.

The same can be observed for other test functions e.g. the Generalized Rosenbrock's test function ($f_{Rosen}(\vec{x}) =$

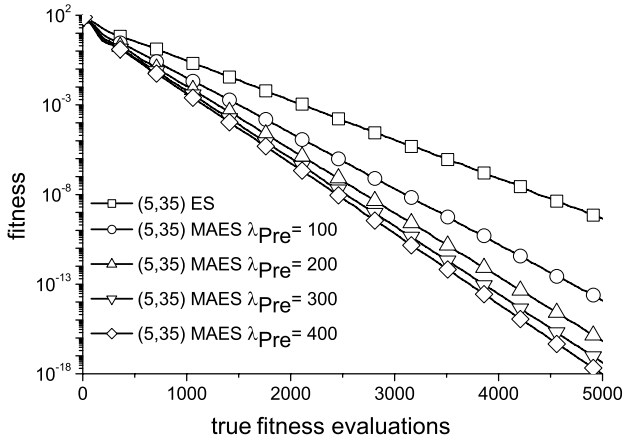

Fig. 2. 10-dim. sphere test function: fitness of best individual for different λ_{Pre} : $100, 200, 300, 400$.

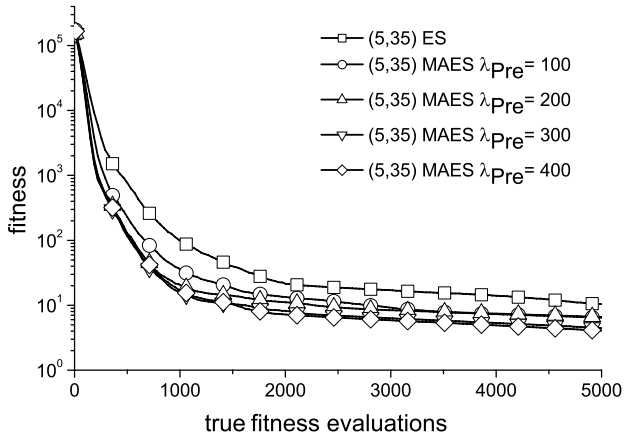

Fig. 3. 10-dim. Generalized Rosenbrock test function: fitness of best individual for different λ_{Pre} : $100, 200, 300, 400$.

$\sum_{i=1}^{10}(100 \cdot (x_{i+1} - x_i)^2 + (x_i - 1)^2)$), see figure (3).

The parameter λ_{Pre} has to be set by the user before the optimization run and keeps constant during the optimization. This fact turns out as an obvious disadvantage of the pre-selection based model assisted algorithm. What is the best value for λ_{Pre}?

It would be very useful if the algorithm itself would select an appropriate value for λ_{Pre} dynamically during the optimization process.

This requirement of model impact control is not new in model assisted evolutionary computation, but there are only a few publications, which discuss this problem.

One approach is to use a confidence criterion given by statistical models like Kriging or Gaussian Processes to control when the approximation model is used instead of the real fitness function [3]. The prediction of the fitness of an individual only substitutes a true fitness evaluation, if the prediction error

given by the Gaussian Process is below a given threshold.

Another approach is given by the adaptive evolution control concept [9]. This approach controls the impact of the model on the evolutionary optimization, through determination of the frequency, when the approximate model is used, based on the model quality. The higher the model quality is, the more often the approximate models are used instead of the true fitness function.

Utilizing the model quality seems to be the most reasonable way to control λ_{Pre}. Therefore we introduce in the following an appropriate model quality criterion, which is later used to control λ_{Pre}.

IV. Model impact control based on model quality: The Controlled Model Assisted Evolution Strategy (C-MAES)

Different quality measures for the approximation model in evolutionary computation are discussed in [8] and [7].

In general the quality of an approximation model is determined by the mean squared error (MSE) of a number N of predictions:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(t_i - f(\vec{x_i}))^2 \tag{6}$$

However, we are not interested in the quality of the model itself, but in the quality of the selection process based on the predictions of the model. In the model assistance framework here used the only purpose of the approximation model is to select the most promising λ out of λ_{Pre} individuals. Therefore the model should only select the best individuals in terms of ranking the fitness predictions.

Although an approximation model with low MSE sorts the individuals regarding their fitness in the correct order, from the evolutionary computation point of view, only the correct selection is of importance [8], [7].

Next we define a quality measure based on the correct model based ranking. The advantage of this quality measure, compared to the MSE is, that it can be easily evaluated for a selection process, which is based on pure random selection. Our considerations are based on the assumption, that a model, which has a better selection quality than a random selection process, supports the optimization process and vice versa.

To measure the quality of the pre-selection process directly, we have to know the true fitness value of all λ_{Pre} individuals. But only the true fitness for the λ most promising pre-selected individuals is known.

But instead of that the quality for a hypothetical selection process can be determined for a selection of μ out of the λ individuals, for which we know the predicted and the true fitness. We assume that this quality measurement is equivalent to the one of the pre-selection process.

A. Selection based model quality

The here described quality measure is a variant of the measures proposed by Jin et al. [8]. The model based selection process selects μ out of the λ individuals with the best predicted fitness. An individual is correctly selected, if it would also be selected by a selection process based on the true fitness of the individuals.

The number of correctly selected individuals gives a quality measure of the model based selection process. Given a rank of $(\lambda - i)$ for each correctly selected individual, if the individual has the $i{-}th$ best fitness based on the true fitness. We define the summed rank of all correctly selected individuals as the quality Q of the model based selection process. For no correctly selected individuals Q is minimal ($Q = 0$). If all individuals are selected correctly, the maximum quality is given as:

$$\begin{aligned} Q^{max} &= \sum_{i=1}^{\mu}(\lambda - i) \\ &= \mu\lambda - \frac{\mu(\mu+1)}{2} \end{aligned} \tag{7}$$

The expectation value of Q for a purely random selection process is given as the product of the expectation value of the number of correctly selected individuals and the expectation value of the rank of a correctly selected individual:

$$\begin{aligned} \left\langle Q^{rand} \right\rangle &= \sum_{i=1}^{\mu} i \frac{\binom{\mu}{i}\binom{\lambda-\mu}{\mu-i}}{\binom{\lambda}{\mu}} \cdot \frac{1}{\mu}\sum_{i=1}^{\mu}(\lambda - i) \\ &= \frac{\mu^2}{\lambda} \cdot \frac{2\lambda - \mu - 1}{2} \end{aligned} \tag{8}$$

B. Controlling the model impact λ_{Pre}

The evolutionary optimization process can only benefit from the model assistance if the selection process performs better and has therefore a better selection quality than a purely random selection process.

After each generation in the ES the actual measured selection quality Q^t has to be compared with the expected quality of the random selection process $\left\langle Q^{rand} \right\rangle$.

For $Q^t > \left\langle Q^{rand} \right\rangle$ the model based selection is better than a random selection and λ_{Pre} should be increased. On the other side, for $Q^t < \left\langle Q^{rand} \right\rangle$ the value λ_{Pre} should be decreased.

Therefore we suggest the following updating rule for λ_{Pre} depending on the actual model selection based quality Q^t. For $Q^t > \left\langle Q^{rand} \right\rangle$:

$$\lambda_{Pre}^{t+1} = \lambda_{Pre}^{t} + \frac{(Q^{max} - Q^t)}{Q^{max} - \langle Q^{rand} \rangle} \cdot \delta_{\lambda_{Pre}} \tag{9}$$

and for $Q^t < \left\langle Q^{rand} \right\rangle$:

$$\lambda_{Pre}^{t+1} = \lambda_{Pre}^{t} - \frac{(\left\langle Q^{rand} \right\rangle - Q^t)}{\langle Q^{rand} \rangle} \cdot \delta_{\lambda_{Pre}} \tag{10}$$

$\delta_{\lambda_{Pre}}$ denotes an adaptation rate. The idea is to compare the actual selection quality of the model with the one of a random process to decide, if there exits a benefit for the evolutionary process by performing model assistance. The bigger the difference, the bigger is the change in λ_{Pre}.

Compared to absolute quality measurements like MSE, the described selection based quality Q has the advantage that one can compare it with $\langle Q^{rand} \rangle$. This is used as a relative quality to control the model influence on the evolutionary process.

V. Experimental results and discussion

To analyze the performance of the algorithms, extensive simulations were performed for real valued test functions. For each test function the following algorithms were compared:

- Standard (μ, λ)-ES
- (μ, λ) MAES with fixed $\lambda_{Pre} = \lambda_{PreStatic} = 2\lambda$
- (μ, λ) C-MAES with controlled λ_{Pre}

The figures show results, which are always evaluated as the mean of 100 repeated runs with different seed values for random number generation. The population size parameter are $\mu = 5$ and $\lambda = 35$.

The size of the pre-selected population for MAES was fixed to $\lambda_{PreStatic} = 2\lambda$ and the C-MAES algorithm with dynamically controlled λ_{Pre} was initialized with $\lambda_{Pre}(t = 0) = 2\lambda$.

A. Standard test functions

The sphere function is a nonlinear, continuous, convex, smooth function, which is an easy test for the self-adaptation mechanism of ES. Both model assisted approaches MAES and C-MAES improve the convergence speed of the standard ES (see figure 4). But C-MAES performs clearly better than MAES.

This result can be explained by the increasing λ_{Pre} during the complete optimization run, due to the good approximation of the fitness landscape of the Sphere function. The selection based quality of the model is always better than the one of the random selection process $\langle Q^{rand} \rangle$. Therefore the evolutionary process profits from the model assistance, which leads to a great enhancement of the convergence velocity.

The same observations are obtained with Schwefel's test function ($f_{Schwefel}(\vec{x}) = \sum_{i=1}^{10} (\sum_{j=1}^{i} x_j)^2$) (see figure 5).

The results for the "Cigar" test function ($f_{Cigar}(\vec{x}) = x_1^2 + \sum_{i=2}^{10} (10000 x_i)^2$) are more interesting. In the first 2000 fitness evaluations of the optimization run the coordinates $2, .., N$ are optimized. In the same way as for the Sphere function λ_{Pre} increases and C-MEAS outperforms ES and MAES. But then the feasible direction along the first coordinate is difficult to find, which is indicated by a plateau in the fitness plot. This is a classical case of a problem, which has only one preferred mutation direction to reach the global minimum. At this point the influence of the model on the optimization process is for C-MAES bigger than for MAES, which results in a faster discovery of the right search direction in objective space.

The Generalized Rosenbrock test function is nonlinear, continuous and not symmetric. The application of model impact control results here only in a slight improvement of the performance of the algorithms (see figure 7), even though λ_{Pre} is increasing, which indicates again a good approximation model.

B. Noisy test functions

In many real world engineering and design optimization applications the fitness function is uncertain or noisy. Noise may result from many different sources, such as measurement errors or numerical instabilities in simulations. To test our approach on such requirements we modify the test functions as follows:

$$F(\vec{x}) = f(\vec{x}) + \delta; \delta \propto N(0, \sigma^2) \tag{11}$$

The objective of optimization on noisy problems is to minimize the expectation value of the fitness function $\langle F(\vec{x}) \rangle$, but the optimization algorithm has only the noisy fitness values $F(\vec{x})$ available.

At the beginning of the optimization runs (see figures 8, 9, 10 and 11) the fitness values are large and the noise is negligible. The fitness landscapes can be approximated by the model with nearly the same quality as in the case without noise. Therefore λ_{Pre} increases and the optimization profits from the model assistance like in section V-A for the test functions without noise.

But with decreasing fitness values the contribution of the noise becomes bigger and it is getting harder to generate an adequate approximation model. Therefore the quality of the model decreases and λ_{Pre} is regulated down. This proves that our adaptation scheme for λ_{Pre} can also react to situations for which no model assistance is desirable.

C. Test functions with dynamically changing noise level

To analyse the ability of C-MAES to react to different model qualities we constructed test cases with iterating noise level. Here the additive noise in the fitness is only switched on between 1000 and 2000 and between 3000 and 4000 fitness calls.

Here the application of the controlled model assistance on the evolutionary optimization process shows the most

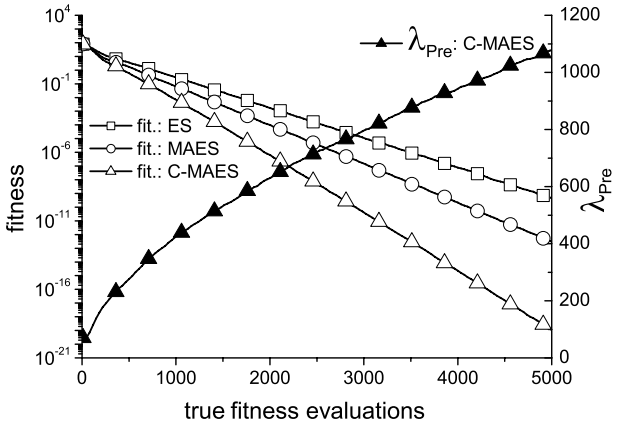

Fig. 4. 10-dim. sphere test function: fitness of best individual and λ_{Pre} of the C-MAES algorithm.

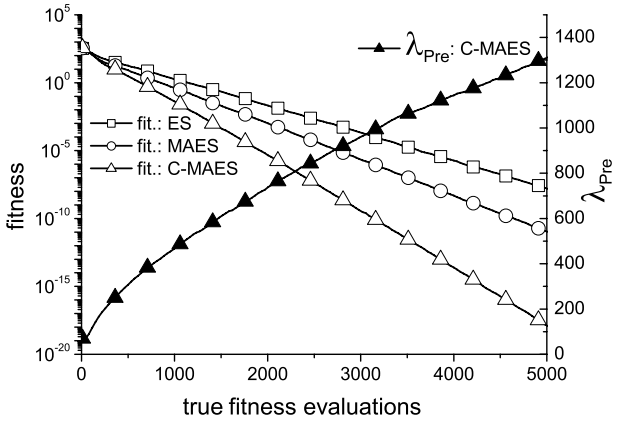

Fig. 5. 10-dim. Schwefels's test function: fitness of best individual and λ_{Pre} of the C-MAES algorithm.

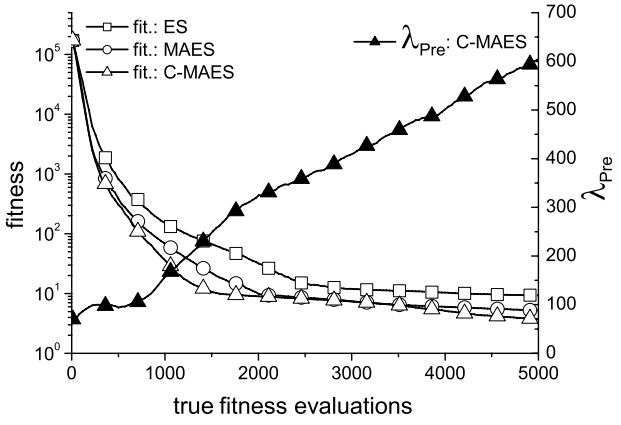

Fig. 7. 10-dim. Generalized Rosenbrock test function: fitness of best individual and λ_{Pre} of the C-MAES algorithm.

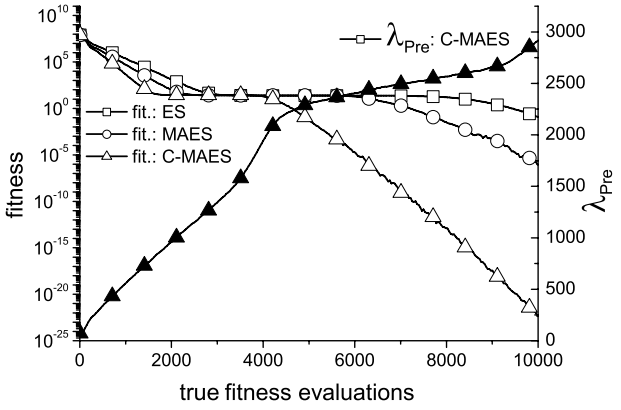

Fig. 6. 10-dim. cigar test function: fitness of best individual and λ_{Pre} of the C-MAES algorithm.

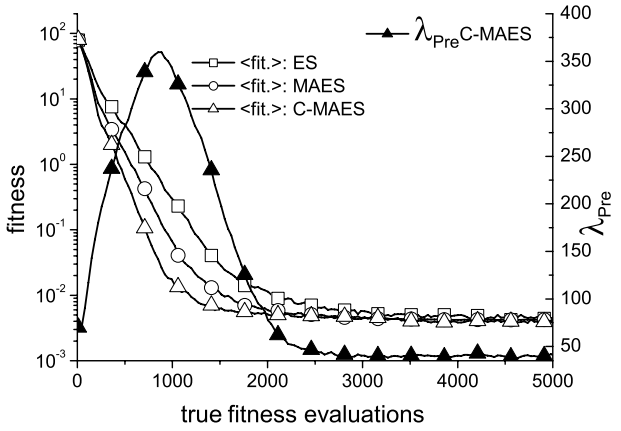

Fig. 8. 10-dim. noisy sphere test function ($\sigma^2 = 0.01$): fitness of best individual and λ_{Pre} of the C-MAES algorithm.

impressive results on the adaptation of λ_{Pre} (see figures 12, 13, 14, 15).

If no noise on the fitness function is present the evolutionary process can benefit from the model assistance and λ_{Pre} increases, which is the same observation as made in section V-A.

After 1000 true fitness calls the additive noise on the fitness function is switched on. C-MAES recognizes that the quality of the model has worsened and decreases the model impact on the optimization process by decreasing λ_{Pre}. Then after 2000 true fitness calls the noise is switched off and again λ_{Pre} increases.

This ability of C-MAES to react to changing qualities of the approximation model, results in a higher convergence rate compared to MAES with fixed λ_{Pre} and the standard ES. Furthermore C-MAES finds better solutions with fewer true fitness calls.

The results show, that the application of model impact control by using an adaptation mechanism for λ_{Pre} enhances the performance of model assisted ES.

VI. Conclusions

We applied a SVR to assist a standard ES by using the model to pre-select the λ most promising individuals from a number of λ_{Pre} individuals. Only these λ are evaluated by the true fitness function.

The application of model assistance enhances the evolutionary optimization process very much. However, the performance of this approach is very sensitive to the value of λ_{Pre}. We proposed a new mechanism, which dynamically controls λ_{Pre} by a model quality measurement based on the number of correctly pre-selected individuals.

Extensive simulations showed that controlling the impact of the approximation model on the evolutionary optimization process enhances the performance of model assisted Evolution

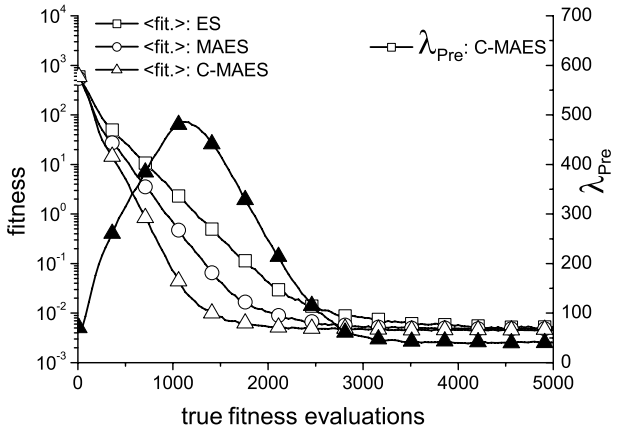

Fig. 9. 10-dim. noisy Schwefel test function ($\sigma^2 = 0.01$): fitness of best individual and λ_{Pre} of the C-MAES algorithm.

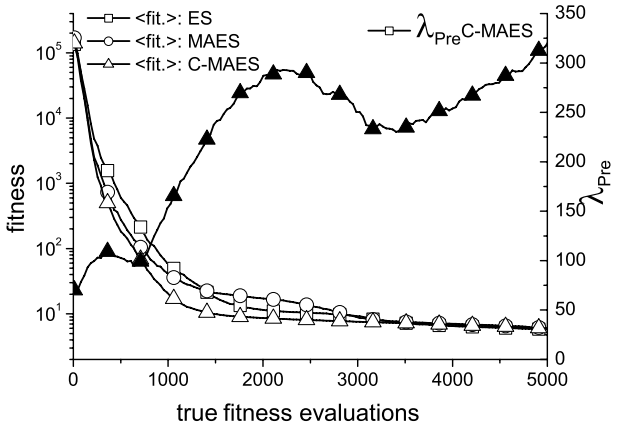

Fig. 11. 10-dim. noisy Generalized Rosenbrock test function ($\sigma^2 = 0.01$): fitness of best individual and λ_{Pre} of the C-MAES algorithm.

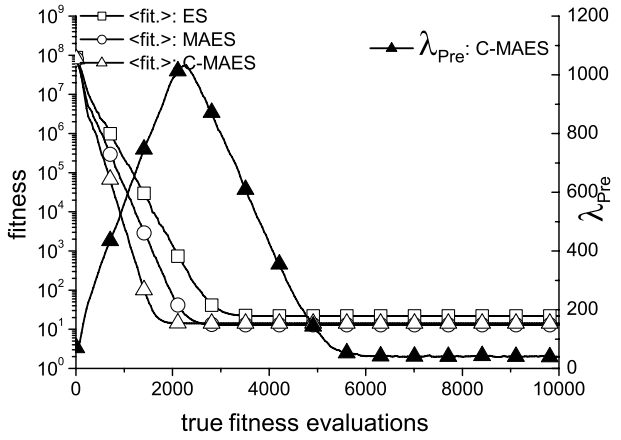

Fig. 10. 10-dim. noisy cigar test function ($\sigma^2 = 0.01$): fitness of best individual and λ_{Pre} of the C-MAES algorithm.

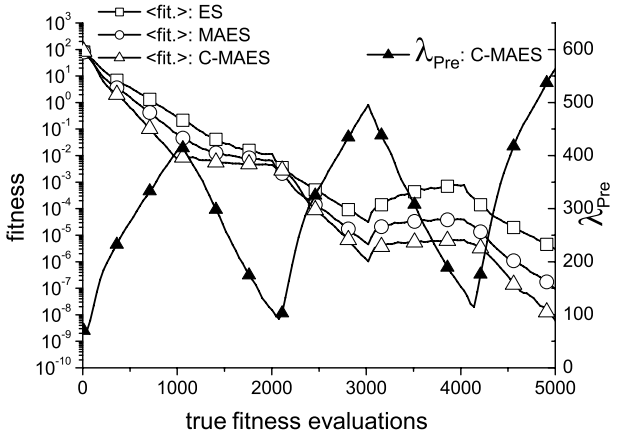

Fig. 12. 10-dim. sphere test function with changing noise level ($\sigma^2 = 0.01$): fitness of best individual and λ_{Pre} of the C-MAES algorithm.

Strategies with fixed model influence. The concept of model impact control could empirically proved for noisy fitness functions with or without time dependent noise level.

If the model has a good quality, which means that it preselects the most promising individuals in the same manner as the actual true fitness function, then the model impact given by λ_{Pre} increases.

On the other side, in areas of the fitness landscape, where a good approximation is not possible e. g. in the presence of a not negligible noise level, λ_{Pre} decreases.

These encouraging results justify the application of model impact control in the field of model assisted ES, which have applications for problems with very expensive and time consuming fitness evaluations like in design optimizations.

For further work it is planned to develop a mechanism, which also takes the costs of model training and evaluation compared to the costs of true fitness evaluations into account.

Acknowledgments: This research has been funded by the German federal ministry of research and education (BMBF) in the project "Entwicklung eines Systems zur automatisierten Herstellung und Charakterisierung von kristallinen Festkörpern in hohem Durchsatz" under contract No. 03C0309E.

REFERENCES

[1] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[3] M. A. El-Beltagy and A. J. Keane. Evolutionary optimization for computationally expensive problems using gaussian processes. In CSREA Press Hamid Arabnia, editor, *Proc. Int. Conf. on Artificial Intelligence IC-AI'2001*, pages 708–714, 2001.

[4] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature VII*, pages 362–370, 2002.

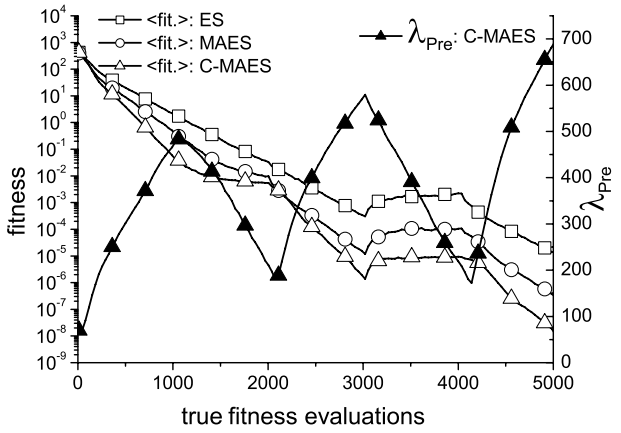

Fig. 13. 10-dim. Schwefel test function with changing noise level ($\sigma^2 = 0.01$): fitness of best individual and λ_{Pre} of the C-MAES algorithm.

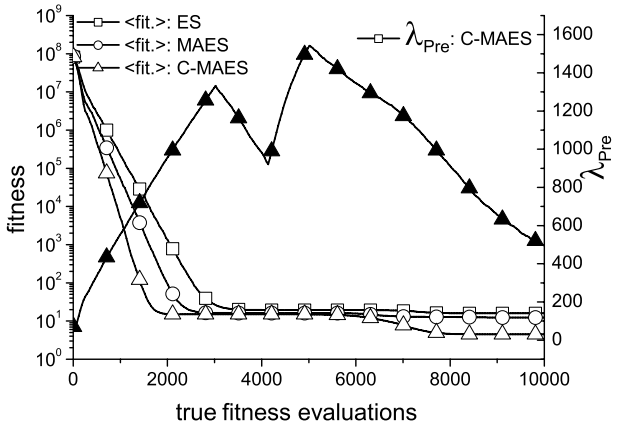

Fig. 14. 10-dim. cigar test function with changing noise level ($\sigma^2 = 0.01$): fitness of best individual and λ_{Pre} of the C-MAES algorithm.

[5] N. Hansen and A. Ostermeier. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_i, \lambda)$-cma-es. In *5th European Congress on Intelligent Techniques and Soft Computing*, pages 650–654, 1997.

[6] Y.-S. Hong, H.Lee, and M.-J. Tahk. Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Engineering Optimization*, 35(1):91–102, 2003.

[7] M. Hüscken, Y. Jin, and B. Sendhoff. Structure optimization of neural networks for aerodynamic optimization. *Soft Computing Journal*, 2003. In press.

[8] Y. Jin, M. Hüsken, and B. Sendhoff. Quality measures for approximate models in evolutionary computation. In Alwyn M. Barry, editor, *GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 170–173, Chigaco, 11 July 2003. AAAI.

[9] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation. March 2002*, pages 481–494, 2002.

[10] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 40(4):687–696, 2003.

[11] J. Poland and A. Zell. Main vector adaptation: A cma variant with linear time and space complexity. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1050–1055. Morgan Kaufman, 2001.

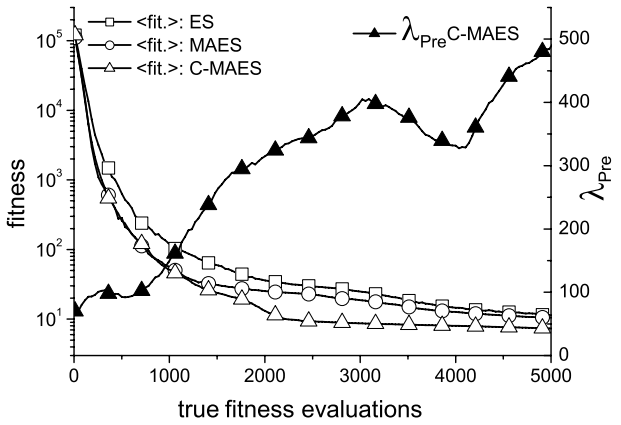

Fig. 15. 10-dim. Generalized Rosenbrock test function with changing noise level ($\sigma^2 = 0.01$): fitness of best individual and λ_{Pre} of the C-MAES algorithm.

[12] A. Ratle. Accelearating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben et al, editor, *Parallel Problem Solving from Nature V*, pages 87–96, 1998.

[13] I. Rechenberg. *Evolutionsstrategie '94*. frommann-holzboog, Stuttgart, 1994.

[14] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser, Basel, 1977.

[15] A. Smola and B. Schölkopf. A tutorial on support vector regression. Technical report, 1998.

[16] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies assisted by gaussian processes with improved pre-selection criterion. In *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, Canberra, Australia*, pages 692–699. IEEE Press, 2003.

[17] H. Ulmer, F. Streichert, and A. Zell. Model-assisted steady-state evolution strategies. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 610–621, Chicago, 12-16 July 2003. Springer-Verlag.

[18] L. Willmes, T. Bäck, Y. Jin, and B. Sendhoff. Comparing neural networks and kriging for fitness approximation in evolutionary optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, Canberra, Australia*, pages 663–670. IEEE Press, 2003.

[19] K. S. Won, R. Tapabrata, and K. Tai. A framework for optimization using approximate functions. In *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, Canberra, Australia*, pages 1520–1527. IEEE Press, 2003.