Ersch. in: Simulation in Industry : 8th European Simulation Symposium 1996, ESS '96 ; October 24 - 26, 1996, Sheraton Hotel and Conference Centre, Genoa, Italy / Agostino G. Bruzzone, Eugene J. H. Kerckhoffs (eds.). - San Diego, CA : The Society for Computer Simulation, 1996. - S. 181-186. - ISBN 1-56555-099-4

Acceleration of Direct Model Optimization Methods by Function Approximation

M. Syrjakow*, H. Szczerbicka^, M.R. Berthold*, K.-P. Huber*

*Institute for Computer Design and Fault Tolerance (Prof. D. Schmid)
University of Karlsruhe, 76128 Karlsruhe,
P.O. Box 6980, Germany

^Department of Computer Science
(Fb3 Informatik)
University of Bremen, 28334 Bremen,
P.O. Box 330440, Germany

KEYWORDS

Simulation, Model Optimization, Direct Optimization, Function Approximation, Neural Networks, Rectangular Basis Function Networks.

ABSTRACT

In recent years optimization of simulation models has become a very important application field of direct optimization strategies. The search process of these iterative strategies is only based on cost function values and does not require any additional analytical information like gradients etc. Today the most common direct methods for global optimization are Genetic Algorithms, Evolution Strategies, and Simulated Annealing. All these methods apply sophisticated probabilistic search operators which imitate principles of nature. Although these operators have been proven to be well-suited for global search the required computational effort (number of required cost function evaluations) still remains a big problem. In this paper we focus on acceleration methods for direct global optimization strategies. Our approach is based on cost function approximation. As approximation techniques we use a simple grid-based method and RecBFNs (Rectangular Basis Function Networks), a special kind of neural networks. The methods we have developed have been applied successfully to model optimization as well as to a selection of mathematical test problems. The encouraging results presented in this paper show that it is possible to optimize simulation models both successfully and with tolerable computational effort.

I. INTRODUCTION

Simulation and especially optimization of complex technical systems has become a research field of great interest in recent years. Because of the enormous increase in computational power it is possible today to combine both disciplines, i.e. to integrate the usually very expensive simulation process into an optimization process. This integration is realized in REMO (**RE**search **M**odel **O**ptimization Package) representing a software tool which enables the modeller to do both, model implementation and model evaluation through a conventional simulation tool, but also optimization of model behaviour. The main difficulties which arise in optimization of simulation-based cost functions are:

- definition of the optimization problem

  Optimization problems can be formalized as a pair (L,F). The solution space L denotes the finite set of all possible solutions. The cost function F is a mapping defined as $F:L \rightarrow R$. In this paper we focus on parameter optimization problems where the search space is a subset of R^n ($L \subset R^n$). The formulation of the cost function F may be very difficult especially if contradictory goals are involved. In this case F has to be composed of several outputs of the simulation model.

- black-box situation

  A simulation model can be viewed as a black-box system because usually there does not exist a mathematical representation for it. Thus no additional analytical information like gradients, etc. is available to accelerate the optimization process of a simulation-based cost function. Here, only direct optimization methods are applicable which exclusively require cost function values.

- expensive model evaluation process

  Evaluation of a simulation model usually requires a lot of computation time which in practice may last from several minutes until many hours or even days. For that reason the optimization process should only require a very limited number of simulation runs to reach the optimization goal.

- stochastic inaccuracies

  If a simulation model comprises stochastic variables its model outputs are distorted by stochastic inaccuracies. For that reason the applied direct optimization methods should be robust against inaccurate cost function values.

- high dimensional search space with complex parameter restrictions

- multimodal cost function with many local and/or global optimum points

To be able to cope with the non-trivial task of model optimization described above we have developed a new class of direct optimization algorithms called *combined 2-phase strategies*. These strategies are based on the splitting of the optimization process into two phases: pre-optimization with a probabilistic global optimization method (Genetic Algorithms (Goldberg 1989), Evolution Strategies (Schwefel 1981), Simulated Annealing (Aarts and Korst 1990), etc.) and fine-optimization performed by a deterministic local Hill-Climber (Pattern Search of Hooke and Jeeves, etc.). The task of pre-optimization is to explore the search space in order to find promising regions where global optimum points might be located. Outgoing from the results of pre-optimization the task of fine-optimization is to localize the optimum point in a promising region efficiently and exactly. So, pre-optimization is predominantly responsible for optimization success, whereas fine-optimization has to ensure optimization quality.

The basic structure of combined 2-phase strategies, their realization within our model optimization tool REMO (REsearch Model Optimization Package) as well as optimization results were already described in (Syrjakow and Szczerbicka 1995). In this publication, we will focus on methods for further acceleration of our developed optimization algorithms.

Within a combined 2-phase optimization strategy the pre-optimization phase, in contrast to fine-optimization, still offers some possibilities to save expensive cost function evaluations. Here the goal is not to achieve high optimization quality but only a rough approximation of a promising region in the search space. Possibilities to reduce computational effort during the pre-optimization phase principally are:

- reduction of simulation run time

  Fine-optimization requires long simulation run times because we want to locate optimal solutions very accurately. During pre-optimization simulation run time can be reduced because here the goal is only to roughly find a promising region.

- avoidance of reevaluation

  Especially if population based optimization algorithms like Genetic Algorithms or Evolution Strategies are applied it often happens that search points are generated several times during the optimization process. Reevaluation of these search points can easily be avoided without any loss of accuracy by search of the optimization trajectory, which comprises all search points already generated and evaluated during the optimization process.

- approximation of the optimized cost function

  This approach is based on the robustness of probabilistic direct global optimization strategies against inaccurately evaluated cost function values. A lot of expensive cost function evaluations can be saved if the cost function is approximated in frequently visited regions of the search space.

In Section II our approach to acceleration of direct global optimization methods by function approximation is outlined in general. Section III describes a simple grid-based technique for function approximation. In Section IV a more sophisticated approximation mechanism based on RecBFNs is explained. Subsequently in Section V some computations results are presented. Finally in Section VI we summarize and draw some conclusions.

II. ACCELERATION OF GLOBAL OPTIMIZATION BY COST FUNCTION APPROXIMATION

Fig. 1 outlines in general how approximation techniques can be integrated into the direct model optimization process. This is done by embedding a function approximator between the direct pre-optimization process and the process of model evaluation performed by a simulation tool. Parameter vectors $\vec{x}$ generated by pre-optimization are not directly passed on to the evaluation process but first to the cost function approximator. Here an approximation criterion (pre-criterion) decides whether the parameter vector $\vec{x}$ is evaluated as usual by simulation or approximated by application of an approximation mechanism. This decision is based on parameter vectors which were already evaluated in previous iteration steps. If approximation is not possible the parameter vector $\vec{x}$ is passed on to the evaluation process which evaluates the corresponding cost function value $F(\vec{x})$.

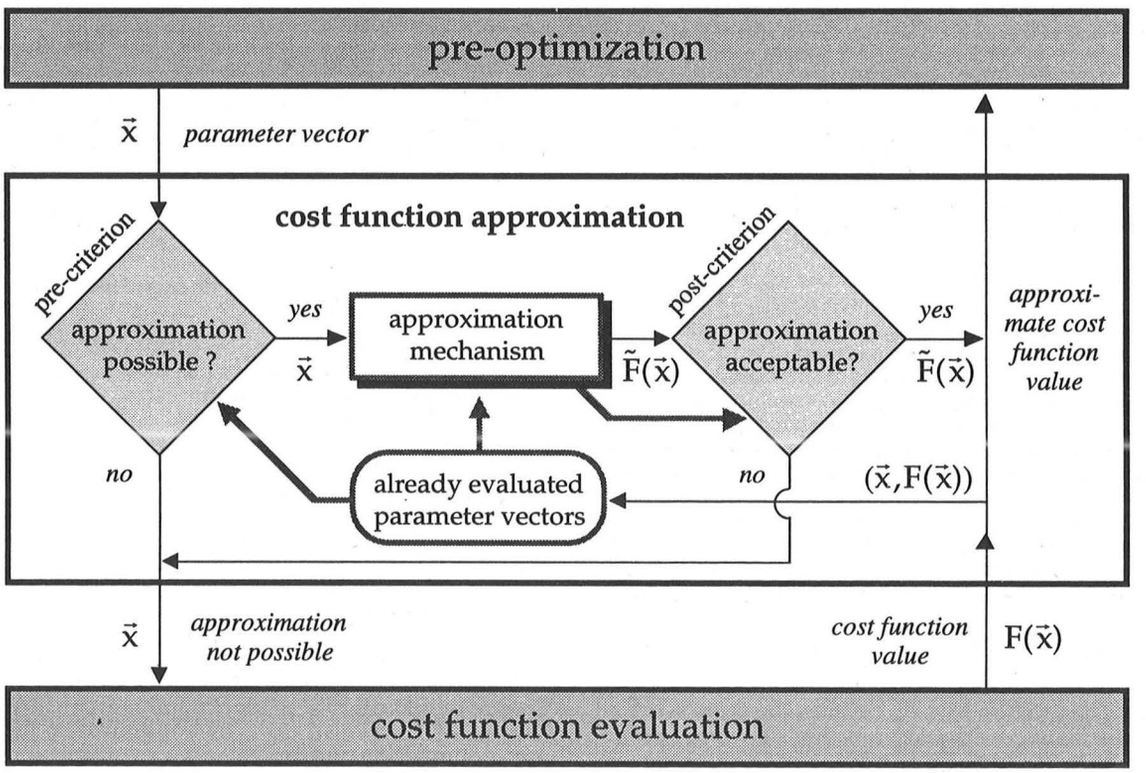

Fig. 1: Efficient pre-optimization by cost function approximation

Subsequently the pair $(\vec{x}, F(\vec{x}))$ is forwarded to the cost function approximator as well as to the pre-optimization process. Along with the progress of pre-optimization the number of already evaluated parameter vectors rises which simultaneously heightens the probability that an approximation is possible. Approximation is performed by an approximation mechanism which calculates an approximate cost function value $\tilde{F}(\vec{x})$. Whether the approximate value is accepted or not decides an acceptance criterion. This so-called post-criterion only makes sense if the approximation mechanism besides the approximate value computes additional data which allows to draw conclusions about the approximation quality. If $\tilde{F}(\vec{x})$ is not accepted an additional cost function evaluation has to be performed. Otherwise a cost function evaluation can be saved because $\tilde{F}(\vec{x})$ is passed on to the pre-optimization process instead of the real cost function value. Especially if the pre-optimization process converges towards a region of the search space a lot of possibly very expensive cost function evaluations can be avoided by cost function approximation.

Finally it should be mentioned that the pre-optimization process does not know whether a delivered cost function value was actually evaluated or only approximated. In the following two sections some possibilities for realization of a cost function approximator are proposed.

III. GRID-BASED APPROXIMATION

The approximation mechanism presented in this Section is based on a partitioning of the search space into sectors of equal size (Syrjakow and Szczerbicka 1993). Let us assume that Genetic Algorithms are applied to pre-optimization and that the search space L is defined through n intervals of real numbers $[a_i, b_i]$; $a_i, b_i \in R$; $i \in \{1,...,n\}$ where $n \in N$ denotes the number of optimized model parameters. Because Genetic Algorithms operate on fixed-length binary strings, these n intervals have to be mapped linearly to integer intervals $[0, 2^{l_i}-1]$, $l_i \in N$, $i \in \{1,...,n\}$. Consequently, the size of the discretisized search space is 2^{cl} with $cl = \sum_{i=1}^{n} l_i$. To partition the search space into sectors of equal size, the n integer intervals $[0, 2^{l_i}-1]$, $l_i \in N$ are mapped again linearly to integer intervals $[0, p_i]$, $p_i \in N$, with $p_i \ll 2^{l_i}-1$. The result is a set of sectors

$$S = \{(s_1, s_2, ..., s_n) \in N^n \mid s_i \in \{0,...,p_i\},$$
$$p_i \in N; i \in \{1,...,n\}; n \in N\}.$$

The number of sectors in which the search space is subdivided can easily be calculated as follows:

$$|S| = \prod_{i=1}^{n} (p_i + 1).$$

The example of Fig. 2 shows a 2-dimensional search space which is partitioned into 81 sectors of equal size using the partition parameters $p_1, p_2 = 8$.

The regular grid shown in Fig. 2 is used to accumulate information for the approximation process. For that purpose during the pre-optimization process for each sector $s \in S$ a statistic is kept, recording

e^s: number of entries into sector s,

F^s_{avg}: average of all cost function values evaluated so far in sector s,

F^s_{max}: maximum of all cost function values evaluated so far in sector s,

F^s_{min}: minimum of all cost function values evaluated so far in sector s.

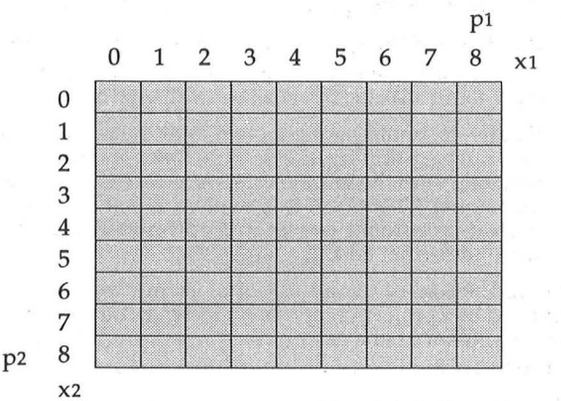

Fig. 2: Partition of a 2-dimensional search space into 81 sectors of equal size

Based on the gathered information accumulated in the grid above it is possible to specify approximation criteria to decide whether a search point $\vec{x}$ can be approximated or not. Such a pre-criterion can be simply defined as follows: When the pre-optimization strategy requires the fitness of a search point $\vec{x}$ in sector s, first the number of previous entries e^s into this sector is checked. If e^s stays below a certain value e^s_{max} (called approximation bound), the cost function is evaluated as usual. Otherwise, a cost function evaluation can be avoided because the approximate cost function value $\tilde{F}(\vec{x}) = F^s_{avg}$ is used for pre-optimization instead of $F(\vec{x})$. In case of maximization (minimization) it is also possible to employ F^s_{max} (F^s_{min}) as the approximate function value. Besides the approximate cost function value $\tilde{F}(\vec{x})$ the simple grid-based approximation mechanism does not deliver any additional data which can be used to estimate approximation quality. For that reason the definition of a post-criterion which decides whether an approximate value is acceptable or not does not make sense in this case.

Applying the grid-based approximation technique, the modeller has to cope with two kinds of parameters. Through the partition parameters p_i it is possible to adjust the number of sectors and their size, whereas the approximation bound e^s_{max} enables to determine how much information should be gathered within a single sector. Now, of course the question arises: "How should we parametrize our grid-based cost function approximator in order to achieve a maximum number of approximations without a considerable loss of optimization success and quality of the pre-optimization result?" This is obviously a very difficult optimization problem which strongly depends on the characteristics of the given cost function. In regard of a successful application of the grid-based function approximator the following hints should be noticed:

- Good heuristic properties can be expected if the search space is partitioned in that way that one sector com-

prises exactly one region of attraction of an optimum point. An example for such an "optimal" partition is given in Section V.

- If there is no a-priori knowledge available about F, the control parameters p_i, $i \in \{1,...,n\}$ and e^s_{max} have to be chosen randomly. However, the partition should not be determined too rough in order to avoid that several optimum points are covered by one sector.
- The choice of the p_i, $i \in \{1,...,n\}$ is closely coupled with the choice of the approximation bound e^s_{max}. In case of a spare partition of L more information should be gathered about a sector than in case of many small sectors.
- Finally, it should be mentioned, that the required memory for partitioning the search space increases exponentially with the number of parameters to optimize (dimensions). Therefore, this method is not applicable to high dimensional parameter optimization problems without further ado.

The reduction of computational effort, which can be achieved by application of this method, mainly depends on the convergence behaviour of the pre-optimization strategy but also on the choice of the control parameters p_i and e^s_{max}. If we partition the search space more sparingly, we certainly have the chance to achieve a better effort reduction, but we have to pay for it with a loss of accuracy.

Some encouraging experimental results achieved with the grid-based approach are presented in Section V.

IV. RecBFN-BASED APPROXIMATION

In contrast to the grid-based approach a rule learning algorithm generates regions of interest only where needed and not a-priori. This can lead to fewer regions and is therefore also applicable for high dimensional problems. The RecBFN (Huber and Berthold, 1995) used in this paper aims to combine the expressive power of Fuzzy Rules with the learning ability of Neural Networks. Using a constructive training algorithm allows to build a network of Rectangular Basis Functions from scratch using training data. The internal structure of the resulting network then enables an interpretation of the resulting network in form of a set of fuzzy rules. The entire system is sketched in Fig. 3. It can be divided into two stages, the Neural Network module and a second unit responsible for fuzzification of training values and defuzzification of the network output.

Training of the network requires classified input patterns. Therefore the fuzzification unit is used to determine which class each pattern belongs to. The network is trained to correctly classify the corresponding input patterns. Function approximation takes the reverse step, the network produces membership values for all classes and the defuzzification unit computes the final output value using the well-known center-of-gravity method.

Rectangular Basis Function Networks can be seen as feed-forward networks with one hidden layer. The input layer is fully connected to the hiddenlayer. Each unit in the hidden layer represents a so-called fuzzy point, i.e. a rectangular area with a trapezoidal membership function. This is achieved through definition of a core-region and a larger support-region. Inside the core-region the degree of membership is equal to 1 and towards the boundary of the support-region this membership value linearly declines to 0.

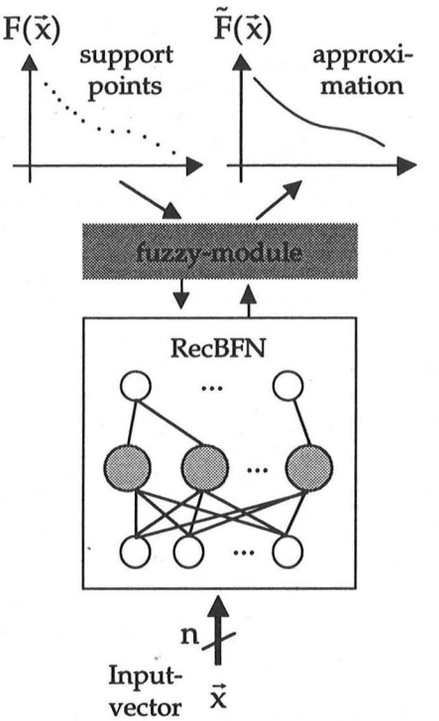

Fig. 3: Function approximation with RecBFNs

The goal of the used training algorithm is to cover all training patterns with the core of at least one RecBF of their corresponding class. In contrast to common classification algorithms not all classes have to be mutually exclusive, instead it is possible to declare a compatibility-relation between different classes, making it possible for patterns of one class to be covered by a support- or core-region of compatible classes. For all other (or incompatible) classes the algorithm guarantees that patterns do not lie inside the support-region of conflicting classes. The compatibility-relation can be used to enable the network to tolerate noisy patterns or small oscillations along class boundaries. Therefore the resulting RecBF units will have cores that just cover all patterns of their assigned class. The core may contain other compatible patterns as well. The support region covers the largest surrounding area, just barely avoiding conflicts with incompatible patterns.

After training the feature space is thus divided into local, rectangular areas that belong to one specific class. In contrast to the grid-based approach these rules are independent of each other, i.e. in areas with low variation of the output variable only few, large rules are necessary to sufficiently model the system. In areas with higher variance, more and smaller rules will be introduced, resulting in a finer granulation. Another interesting feature, especially for the application presented here, is the possibility of these rules to ignore irrelevant features. The RecBFN approach is therefore also feasible for higher dimensional search spaces, where the normal grid-based approach fails due to its high computational burden. In addition the used algorithm see (Berthold and Huber 1995) for details does not require any parameters to be adjusted a-priori. The fast training- and approximation-speed allows the user to concentrate on other issues of simulation, rather than spending precious time adjusting sensible parameters that heavily influence the approximators performance.

V. RESULTS

Our developed optimization algorithms have been already applied to several simulation models as well as to a heterogeneous set of mathematical test functions. In this Section, some optimization results achieved with a combined 2-phase strategy are presented. For pre-optimization we use a Genetic Algorithm which is accelerated by a cost function approximator. As approximation techniques the grid-based method as well as RecBFNs are applied. Fine-optimization is performed by the Pattern Search Algorithm of Hooke and Jeeves which represents a very efficient direct Hill-Climbing method. The combined 2-phase optimization algorithm is used for *single-stage* optimization within a *multiple-stage* optimization algorithm. Multiple-stage optimization means that a single-stage method is executed several times successively in order to generate a sequence of optimum points. To avoid that previously found optimum points are located again in subsequent optimization stages a method called avoidance of reexploration (AR) is applied. A detailed description of multiple-stage optimization can be found in (Syrjakow and Szczerbicka 1995). Especially multiple-stage optimization makes the application of function approximation very advantageous. Here with each additional optimization stage the approximation mechanism gets more information about the cost function which can be exploited in previous optimization stages.

Table 1 shows the mathematical test problem which is considered in this paper.

search space	$L^n = \left\{ \vec{x} \in R^n \middle\| 0 \le x_i \le 3 \cdot \pi;\ i \in \{1,\ldots,n\} \right\}$
cost function	$F^n(\vec{x}) = \left\| \prod_{i=1}^{n} x_i \cdot \sin(x_i) \right\|$

Table 1: Mathematical test problem (L^n, F^n)

The multiple-stage optimization strategy was applied to localize the maximum points of (L^n, F^n); $n \in \{2,\ldots,10\}$. Fig. 4 shows a graphical 3D-representation of the 2-dimensional cost function F^2. For all problem dimensions $n \in N$ this function comprises exactly one global optimum point $\vec{x}^*$. The number of local optimum points $\vec{x}^\wedge$ raises exponentially with an increase of the problem dimension n.

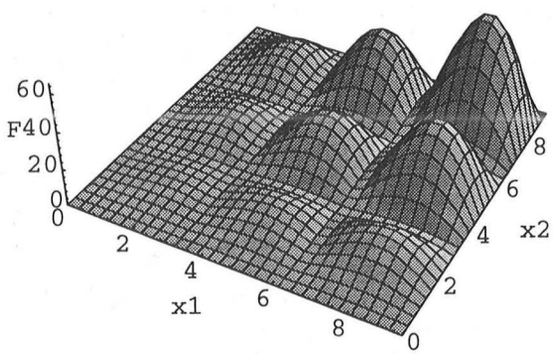

Fig. 4: Graphical 3D-representation of F^2

Before starting our optimization algorithm, we have to determine its control parameter values. For the Genetic Algorithm, we mainly chose standard control parameter settings recommended in literature (Schaffer *et al.* 1989). In the following, the most important control parameter values are presented: For all problem dimensions $n \in \{2,\ldots,10\}$ the crossover rate is set to 0.75, and the mutation rate is set to 0.075. The length of a parameter segment in the fixed-length binary strings (individuals of a population), representing parameter vectors is set to 10 bit. The population size pg (number of individuals in a population) and the control parameter t for the termination criterion are set dimension dependent (see Table 2). Switch-over to fine-optimization occurs if the Genetic Algorithm can not improve the cost function by 5% within t generations.

n	2	3	4	5	6	7	8	9	10
pg	20	20	20	30	30	30	40	40	40
t	2	3	4	4	5	6	6	7	8

Table 2: Dimension dependent control parameter settings of the Genetic Algorithm

After pre-optimization the Pattern Search Algorithm is applied in order to achieve high optimization quality. The demanded accuracy of the Pattern Search Algorithm is set to 0.01.

To reduce the number of cost function evaluations required from the Genetic Algorithm the grid-based as well as the RecBFN-based cost function approximator are applied. The employed control parameter values of the grid-based approximator are summarized in Table 3.

•	cp_1	cp_2	cp_3
$[p_i, e^s_{max}]$	[2,7]	[8,5]	[14,3]

Table 3: Control parameter settings of the grid-based function approximator

In the following the results of 9 optimization experiments with the grid-based approximator are described. For each optimization problem (L^n, F^n), $n \in \{2,3,4\}$ an extensive optimization experiment with the control parameter settings cp_j, $j \in \{1,2,3\}$ shown in Table 3 is performed. Each optimization experiment comprises 200 independent optimization runs with the multiple-stage optimization algorithm. Every multiple-stage run generates a sequence of 20 optimum points. Hence, in one optimization experiment altogether 4000 optimum points are localized.

Diagram 1 summarizes the results of the 9 optimization experiments with the grid-based cost function approximator. The x-axis shows the problem dimension n. The columns represent the average number of cost function evaluations *cfe* required in one optimization stage. The lines show the optimization success *os* achieved in the first optimization stage

$$os = \frac{\text{number of global optimum point hits}}{\text{number of optimization runs}}.$$

Diagram 1 impressively shows that the control parameter stetting cp_1=[2,7] causes an optimal partition of the search space L^n. Using this partition every optimum point of (L^n, F^n) is covered by exactly one sector. Consequently for all considered problem dimensions $n \in \{2,3,4\}$ the highest savings of cost function evaluations could be achieved. With every refinement of the grid the savings go down respectively. As expected, the loss of optimization success rises with an enlargement of the grid sectors. However, regard-

ing the amount of savings the loss of optimization success remains tolerable. Additionally, Diagram 1 points out clearly that with an increase of the problem dimension a good partition of the search space becomes more and more important.

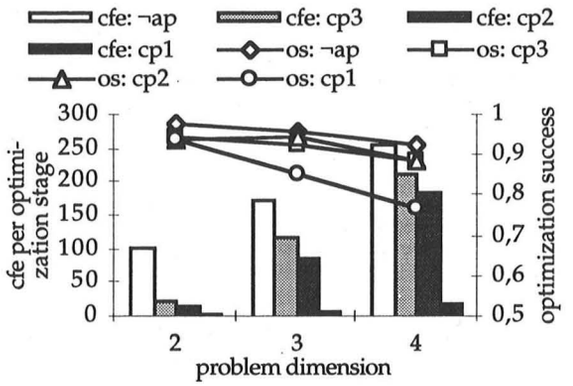

Diagram 1: Optimization results of the grid-based function approximator (¬ap: without cost function approximation)

As mentioned before the grid-based approximation method is only applicable to low-dimensional optimization problems. To solve the higher dimensional problems $n \in \{5,...,10\}$ the RecBFN-based approximator has to be applied. The optimization results achieved with the RecBFN-based cost function approximator are summarized in Diagram 2. As already indicated in Diagram 1 with every increase of the problem dimension it becomes more and more difficult to obtain high savings without a substantial loss of optimization success. At problem dimension 10 good results can only be achieved if the RecBFN is perfectly adapted to the range of the cost function and well trained in the region of attraction of the global optimum point. The increasing deterioration of the results presented in Diagram 2 clearly shows that the application of approximation methods to high dimensional problems is not a trivial task. It also shows that our investigations in that direction are far from being completed. However, we assume that there still exists a great potential for further improvement of our RecBFN-based approximation technique. In order to corroborate this assumption we are intensifying our research work at the moment by developing methods which dynamically adjust initial RecBFN parameter settings optimally to a given optimization problem.

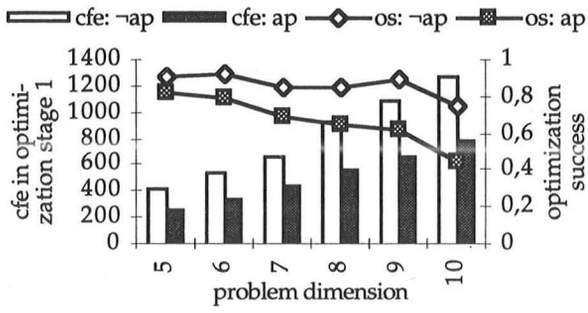

Diagram 2: Optimization results of the RecBFN-based function approximator (ap: approximation with RecBFN)

Summing-up, the optimization results presented above show that pre-optimization can be considerably accelerated by cost function approximation without a substantial loss of optimization success. Because after pre-optimization fine-optimization is applied there is also no loss of optimization quality. In regard of good control parameter settings of the function approximator it is very important to exploit all available a-priori knowledge about the optimization problem.

VI. CONCLUSIONS

In this paper an approach to acceleration of direct global optimization strategies by function approximation was presented. As approximation techniques a grid-based as well as a RecBFN based-method were presented. The simple grid-based technique can be successfully applied to low-dimensional optimization problems. The more sophisticated RecBFNs represent an efficient alternative in high-dimensional search spaces. The optimization results presented in this paper show that it is possible to accelerate pre-optimization without a substantial loss of optimization success. If a combined 2-phase optimization strategy is applied optimization quality is not affected because of fine-optimization.

The topics of our future work concern fine-tuning of our approximation techniques especially cost function approximation with RecBFNs. Beyond that we examine methods for further exploitation of the information which is gathered by the function approximator during the optimization process.

ACKNOWLEDGEMENTS

We want to thank Prof. D. Schmid for his encouragement and support of our work. We also thank our students, especially J. Gramlich and A. Kehl for their engagement, contributions, and implementary work.

REFERENCES

Aarts, E. and J. Korst. 1990. *Simulated Annealing and Boltzmann Machines*. Wiley.

Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Grefenstette, J.J. and J.M. Fitzpatrick. 1985. "Genetic Search with Approximate Function Evaluations." In *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, July 24-26, 1985, Carnegie-Mellon University, Pittsburgh, PA, USA, pp. 112-118.

Huber, K.-P. and M.R. Berthold. 1995. "Building Precise Classifiers with Automatic Rule Extraction." In *Proceedings of the IEEE International Conference on Neural Networks ICNN'95*, Perth, Western Australia, 1995, Vol. 3, pp. 1263-1268.

Schaffer, J.D.; R.A. Caruana; L.J. Eshelman; and R. Das. 1989. "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization". In *Proceedings of the Third International Conference on Genetic Algorithms*, George Mason University, pp. 51-60.

Schwefel, H.-P. 1981. *Numerical Optimization of Computer Models*. Wiley.

Syrjakow, M. and H. Szczerbicka. 1993. "Efficient Model Optimization with REMO". In *Proceedings of the Fifth Workshop on Neural Networks WNN93/FNN93*, November 7-10, 1993, San Francisco, California, pp. 101-106.

Syrjakow, M. and H. Szczerbicka. 1995. "Simulation and Optimization of Complex Technical Systems." In *Proceedings of the 1995 Summer Computer Simulation Conference SCSC'95*, July 24-26, 1995, Ottawa, Ontario, Canada, pp. 86-95.