Reports of the Department of Mathematical Information Technology
Series B. Scientific Computing
No. B 19/2008

Decreasing Computational Cost of Simulation Based Interactive Multiobjective Optimization with Adjustable Solution Accuracy

Timo Aittokoski Kaisa Miettinen

University of Jyväskylä 2008

Reports of the Department of Mathematical Information Technology
Series B. Scientific Computing

Editor Raino A. E. Mäkinen
Technical editor Paula Takala

Reports of the Department of Mathematical Information Technology
Series B. Scientific Computing
No. B 19/2008

Decreasing Computational Cost of Simulation Based Interactive Multiobjective Optimization with Adjustable Solution Accuracy

Timo Aittokoski Kaisa Miettinen

University of Jyväskylä
Department of Mathematical Information Technology
P.O. Box 35 (Agora)
FI–40014 University of Jyväskylä
FINLAND
fax +358 14 260 2731
http://www.mit.jyu.fi/

URN:ISBN:978-951-39-9035-0
ISBN 978-951-39-9035-0 (PDF)
ISSN 1456-436X

Jyväskylän yliopisto, 2022

ISBN 978-951-39-3353-1
ISSN 1456-436X

Decreasing Computational Cost of Simulation Based Interactive Multiobjective Optimization with Adjustable Solution Accuracy

Timo Aittokoski* Kaisa Miettinen†

Abstract

Solving real-life engineering problems can be time-consuming and difficult because problems may have multiple conflicting objectives, functions involved highly nonlinear and containing multiple local minima, and function values are often produced via a time-consuming simulation process. Problems of this type can be solved using global multiobjective optimization methods, preferably with interactive approaches, which allow the designer (or decision maker in general) to learn about the behaviour of the problem during the solution process.

In an interactive approach the designer specifies preferences and Pareto optimal solution(s) following these preferences are generated, typically by forming a scalarizing function and solving it. In simulation based optimization this may take time. Thus, the designer may have to wait for a long before (s)he can continue the solution process. Although some efficient global optimization algorithms exist, it is of outmost importance to be able to reduce the computational burden.

In our study, we show that substantial savings in calculation time can be achieved using a decreased number of function evaluations at the beginning of the interactive solution process, without compromising the quality of the final solution too much. Furthermore, at each iteration we use simple heuristics to judge sufficient amount for computation. As the designer has gained more understanding about the problem, (s)he may approach the final solution with an ever increasing accuracy and number of objective function evaluations. We show results using several different budget schemes for calculation, and identify levels where a sufficient quality for final solutions is retained.

*timo.aittokoski@jyu.fi

†kaisa.miettinen@jyu.fi

Department of Mathematical Information Technology, University of Jyväskylä, PO Box 35 (Agora), FI-40014 University of Jyväskylä, Finland

1 Introduction

Behavior of many real world systems and devices can often be expressed using specific computer implemented mathematical models, simulators. In industry, simulators are often used instead of more concrete appliances, since running the simulator is usually far cheaper, faster and in some cases also safer than real world prototyping. Although simulators allow, for example, a designer to try multitude of different designs and explore their advantages and disadvantages, they give no information about how exactly that behavior can be improved. For this reason, an advanced design protocol should employ optimization algorithms to systematically explore different design variable combinations.

When an optimization system is build on top of a simulation software, this poses some special requirements on the optimization algorithm. Objective function evaluations can be computationally very expensive, as the execution time of a few minutes for one single simulation run can be considered normal. Also partial derivatives which are commonly used to guide (local) optimization processes are usually unavailable as the output data of the simulation software is often the result of a complex sequence of calculations. The simulator may be a black box when not even automatic differentiation can be applied. Further, the objective function itself may have lots of locally optimal values, and there often are several conflicting objectives (instead of only one objective function) that should be considered at the same time. All these facts suggest that for general simulation based optimization systems, one should use efficient (in terms of objective function evaluations) global optimization algorithms in a multiobjective manner.

In multiobjective optimization, we can identify compromise solutions, so-called Pareto optimal solutions (where, by definition, none of the conflicting objectives can be improved without impairing at least one of the others), and the multiple objectives are often handled by converting them into a problem having a single objective function (see, e.g., [31]). These, so-called scalarization functions, typically also incorporate designer's preference information about the conflicting objectives, that is, what kind of values of objectives are desirable. By solving the scalarized problem with an appropriate single objective solver, we get Pareto optimal solutions to the original problem. In case of nonconvex problems, depending on the properties of the optimization algorithm used (i.e., local or global), the resulting solutions are either locally or globally Pareto optimal.

In interactive multiobjective approaches, the most satisficing Pareto optimal solution is looked for so that the designer can adjust preference information stepwise and at the same time learn about the problem characteristics. This is important because the designer does not necessarily have profound understanding about the behavior of the complete problem at the beginning of the solution process, or in other words, what kind of solutions can be reached and which ones are unattainable. In an iterative and interactive solution process, the designer directs the search and only those Pareto optimal solutions are generated that are interesting to him/her.

With simulation based optimization problems, the computational complexity of

the objective functions may hinder the solution process. To cope with computational complexity, it is necessary to use algorithms with as high efficiency as possible, i.e. algorithms which produce good objective function values using as few objective function evaluations as possible. On the level of global optimization algorithms, several improvements to efficiency have been made, for example, using response surface methods. Our approach is somewhat different, as we are dealing with the efficiency of the whole interactive optimization process, instead of only efficiency of the particular optimization algorithm.

In this study we present a new approach to aim at computational efficiency with regard to interactive, simulation based multiobjective optimization, and we show, that at least in some cases, substantial savings in calculation time can be achieved using a decreased number of function evaluations at the beginning of an interactive solution process, without compromising in the quality of the final solution too much. We show results using several different budget levels, and show that even rather small amounts of function evaluations may be sufficient at the beginning of the optimization procedure.

Particularly, we aim at improving the computational efficiency of a whole interactive simulation based optimization system, which we have earlier introduced in [1] and [2]. We strive to accomplish this by exploiting the well-known fact that many global optimization algorithms are stochastic by nature, and thus the quality of the solution improves as the number of objective function evaluations is increased. It is also known that at the beginning of an interactive optimization process, the designer is many times also at a beginning of the learning curve: (s)he is learning of how different objectives are interrelated, what the trade-offs between the objectives are and, also, what kind of objective values can be reached in general. Thus, at the beginning of the optimization process, quite a coarse accuracy may be sufficient, and thus only a small amount of objective function evaluations is needed to get the grasp of the problem. During the interactive solution process we adjust the computational accuracy of the optimization algorithm by stopping each of the iterations when a sufficient number of objective function evaluations has been reached. To help judge what is a sufficient number of objective function evaluations, we introduce a maximum difference percentage (MDP) measure.

As the designer has gained more understanding about the problem, (s)he may approach the final solution with an ever increasing accuracy and number of objective function evaluations. Ultimately, the final solution can be calculated using a MDP threshold that assures the designer of a good enough solution quality. We demonstrate these ideas with a problem related to the optimal design of a two-stroke engine expansion chamber. Even though the ideas presented are not limited to design problems only, we here call the decision maker involved in the solution process as a designer.

The rest of this study is organized as follows. In Section 2 we shortly describe an overview of a simulation based optimization system. In Section 3 we give some general references about multiobjective optimization, and we also introduce the interactive NIMBUS method used in this study. We introduce our interactive method

with increasing accuracy in Section 4. In Section 5 we present our example case of internal combustion engine design with three objective functions. Section 6 summarizes results of numerical tests, and in Section 7 we discuss some findings concerning the optimization system. Finally, in Section 8 we draw some conclusions.

2 Simulation based optimization

When considering any optimization task, the problem must first be identified and modelled as an optimization problem. This means that some properties of the system studied must be identified and selected to be improved, i.e., optimized. These properties are referred to as objective functions. They depend on a set of design variables. Typically, we also have constraint functions defining feasible values for the design variables. Feasible design variable values define a so called search space for the particular problem.

From this setting, an optimization algorithm strives to improve (possibly scalarized) objective function value by altering design variable values systematically. When the objective and/or constraint function values are derived from a simulator output, we have a simulation based optimization system in question. In this section we briefly discuss essential elements of a simulation based optimization system and then one part involved, that is, global optimization.

2.1 System overview

While speaking of a simulation based optimization system, it is reasonable to distinguish two main parts of the optimization system, namely the optimizer (the optimization algorithm itself) and the part which calculates values for the objective functions, the simulator (together with some additional software). With analytic problems, which can be expressed in a closed form, one can sometimes use one homogenous system, i.e., the optimization algorithm and objective function calculation can be implemented using the same programming language, and they can even reside in the same executable file. While speaking of more general cases, and especially simulation based optimization, the whole system evidently becomes very heterogenous and consists of several modules, which may be implemented using different tools and languages. In general, the whole system can be divided into four separate modules: optimizer, input interface from the optimizer to the simulator, simulator software, and the output interface from the simulator to the optimizer.

The optimizer module guides the whole optimization procedure by deciding what design variable values should be passed forward. The second module, the input interface for the simulator, receives design variable values and generates from those values suitable input configuration files for the third module, the simulator. For example, for an engine simulator, the engine design variables must be converted to depict the whole internal engine geometry, or some more specific part of the engine, which is to be optimized. To the optimization system the simulator is seen

as "black box": it merely receives system configuration data which reflects current design variable values and produces simulation output files which contain detailed information on how the simulated system performed with a given system configuration. The simulator itself may be an arbitrarily complex set of calculations and even consist of several software modules. The execution time for a single simulation run may vary from milliseconds to weeks.

In the fourth module, output files of the simulator must be handled to constitute values for the objective functions, which are finally passed back to the optimization algorithm. Then the whole loop starts all over again, and the optimization algorithm uses information of the objective function value to decide new values for design variables. The iterative process is continued until the optimizer meets some pre-defined stopping criterion, for example, until an allocated number of function evaluations has been used, i.e. the budget for objective function evaluations is exhausted.

As can be seen, the whole optimization system may be very heterogeneous as each of the four modules may be implemented using different programming languages and platforms and they may even run on physically separate computers. Regardless of the implementation and structure of the modules, they must interface with each others seamlessly. For further details of the four modules, see [2].

2.2 Global optimization in brief

As motivated in the introduction, we often need global methods for optimization problems because the objective function(s) may be nonconvex, i.e., have several locally optimal values, and any local optimization algorithm is able to find only the closest one to the given starting point. In this way, it is very easy to miss the real optimum for the problem at hand using only local algorithms. As many real life engineering problems require by their very nature global solvers, only these are in the scope of our interest.

On the other hand, global optimization algorithms aim at determining the best local optimum among all the local optima in the search space. In the field of local optimization, it is easy to judge when the optimum has been found by checking whether the solution satisfies optimality conditions or not, see, e.g., [16]. Unfortunately, there exists no such general criterion for asserting that the real global optimum has been found. Furthermore, in case of global optimization, there is no information similar to the gradient information, that could be efficiently used to locally decide where to search next. Thus, many global optimization algorithms are stochastic by nature, that is, the probability of finding the global optimum increases when the optimization process is continued further. In other words, this usually means that the more objective function evaluations are used, the better will be the solution gained. This is one feature we utilize here to reduce the number of required objective function evaluations.

A plethora of global optimization methods has been suggested in the literature. For a thorough discussion of global optimization algorithms in general, see, for ex-

ample, [21, 22, 38, 47]. For this study we chose to use two population based optimization algorithms, namely the controlled random search (CRS) [40], and especially the CRS2 [5] variant of it, and differential evolution (DE) [46], where the population is expected to concentrate around the global minimum during the optimization procedure. The use of these is based on the comparisons made in [1], [4] and [44].

In the CRS method, the search space is initially sampled randomly to form a population P. In each of the following steps, a new trial point is generated, and if the objective function value of the trial point is better than the current worst point in P, the worst point is replaced in the population by the new trial point. By this repeated process, points in the population are expected to concentrate around the global optimum. The ideas of the basic CRS algorithms have been further extended, for example, in [5].

The differential evolution is a simple stochastic optimization algorithm. The essence of the DE algorithm is a process for generating trial parameter vectors to be used in mutation. In DE, the weighted difference of two member vectors of the population is added to a third vector. With this scheme, there is no need to use separate probability distribution to create new trial vectors, and this makes the process completely self-organizing, because each dimension of the problem will evolve proportionally over time, taking small steps when the variation in the values of a given design variable within a population is small, and large steps when that variation is large.

Next, we pay attention to another element in the optimization module of a general simulation based optimization system, that is, multiobjective optimization.

3 Multiobjective optimization

Multiobjective optimization is needed whenever there are several conflicting objective functions to be optimized simultaneously, as the case is with many real-life engineering problems. A general form of a multiobjective maximization problem is

$$\begin{array}{ll} \text{maximize} & \{f_1(x), f_2(x), \ldots, f_k(x)\} \\ \text{subject to} & x \in S \end{array} \tag{1}$$

involving k (≥ 2) conflicting objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$. Here, we have n design variables $x \in \mathbb{R}^n$ and $S \subset \mathbb{R}^n$ stands for the feasible design variable space, i.e., search space defined in a general case by inequality and equality constraints. An objective vector $z = (f_1(x), f_2(x), ..., f_k(x))^T \in \mathbb{R}^k$ consists of k objective function values depending on the design variable vector x. Without loss of generality, we restrict our consideration to maximization because if we need to minimize f it is equivalent to maximize $-f$.

In multiobjective optimization, the concept of optimality is not as straightforward and unambiguous as it is in the single objective case. In the multiobjective case, we want to optimize the values of several objectives at the same time, but usually there exists no single point within the feasible design variable space where all

the objectives reach their optima. Because of this conflict, we have a set of optimal compromise solutions. This set of optimal solutions is called a Pareto optimal set. A solution belongs to the Pareto optimal set if none of the objective function values can be improved without impairing the value of at least one other objective.

Because the solutions in the Pareto optimal set cannot be completely ordered mathematically, some additional information is needed to select one of them as the final, most preferred, solution. This information, called preference information, is given by a decision maker or a designer who is supposed to have expertise of the problem domain and be able to express preference information about what kind of solutions are preferred.

Multiobjective optimization problems are often solved by converting the multiple objectives together with the preference information into a single objective optimization problem using so-called scalarizing functions [31]. Scalarizing functions are constructed so that solutions produced by them are Pareto optimal. For many scalarization methods, as well as for designers, some information about the ranges of points in the Pareto optimal set is needed. The upper bounds are defined by an ideal objective vector z^*, whose components are obtained by maximizing each of the objective functions individually. A vector strictly better than z^* is called a utopian objective vector z^{**}. The nadir objective vector z^{nad} consists of worst objective function values in the Pareto optimal set. Typically, it can only be estimated [31].

The scalarized single objective optimization problem can be solved using any appropriate algorithm, and it is very important to notice that if the problem is nonconvex, depending on the type of the optimization algorithm, results are either globally or locally optimal and, thus, Pareto optimal. Local Pareto optimality means that a solution is Pareto optimal only in some small region near that particular solution, instead of the whole search space. This leaves the decision maker into an unintuitive state, as there may exist essentially better (global) solutions in the search space. In other words, settling for locally Pareto optimal solutions violates the idea of Pareto optimality. Thus, in order to get globally Pareto optimal solutions for nonconvex problems, we must use a global single objective optimizer. For a survey of multiobjective optimization methods see, for example, [30], or for a more thorough discussion including different scalarizing functions, see [31] and references therein.

3.1 Computational efficiency in multiobjective optimization

As mentioned above, scalarizing functions can be used to generate one Pareto optimal solution at a time. Methods of this type may be used in an interactive fashion, gaining one solution at a time, then letting the designer adjust the preference information based on the current solution, and solving the new scalarized problem again. Some scalarizing based approaches also produce several solutions for the designer to compare at a time.

The most obvious way to improve the efficiency of scalarization based methods is by improving the algorithmic efficiency, i.e., using more efficient solvers with regard to the amount of function evaluations or absolute calculation time. As motivated

earlier, we here consider global optimization only. There exist several global optimization algorithms which strive for efficiency, for example DIRECT [39], Multilevel Coordinate Search (MCS) [23], GROPE [11], Differential Evolution (DE) [46], Controlled Random Search (CRS) [40] (and its variants) [5], Recursive Random Search algorithm [51], Efficient Global Optimization (EGO) [24], a variant of it, ParEGO [25], an enhancement of EGO called SuperEGO [42], hybrid methods Simplex Coding GeneticAlgorithm (SCGA) [18] and the Simulated Annealing Heuristic Pattern Search method (SAHPS) [19].

Efficiency for some of the abovementioned methods is reported in [4], [18], [19] and [44]. Furthermore, a comparison between DE, CRS, SuperEGO, SCGA and SAHPS was conducted in [1].

Another approach for solving multiobjective optimization problem, known in the literature as a posteriori methods or approximating methods [31], produce a discrete approximation and representation of the whole Pareto optimal set. In this case, no preference information is needed before the representative set of Pareto optimal solutions has been generated, and the decision maker can decide afterwards which solution satisfies him/her the most.

By their very nature, these methods, like evolutionary algorithms [9, 53], tend to lack computational efficiency, although some attempts towards better efficiency have been made, e.g. in [3, 13, 43]. For a review of the approximation methods, see, for example, [41].

Another problem with these methods lies in the selection of the most preferred, the final solution among the set of nondominated solutions. An usual way to accomplish the selection of the final solution is by visualization of the resulting Pareto optimal set; though, it is innate only in the case of two objective functions. With three objective functions, some visualizations can be produced, for example, as projections to two dimensions, but with four or more objectives, intuitive and easily understandable visualization is practically impossible. However, without such tools it is really hard for the designer to select the most preferred solution among dozens of k-dimensional objective vectors forming the representative set. In our study we want to retain the efficiency and expandability of the system to more than only two or three objective functions.

As approximating methods aim at producing an approximation of the whole Pareto optimal set, it is intuitively obvious that a great majority of these solutions is not of interest to the decision maker, as (s)he is looking for some very certain compromise between the objectives. As a direct consequence of that, lots of computational effort may be wasted to produce uninteresting solutions. And as already mentioned, representing the plethora of multidimensional Pareto optimal solutions to the decision maker may often surpass his/her cognitive capacity, and the selection of the final solution from the big set may become a problem in itself.

For the reasons mentioned so far, we focus in this study at interactive methods, where only those solutions of the Pareto optimal set are generated that the decision maker considers appealing.

3.2 The NIMBUS method

From our point of view, an interesting type of methods is interactive methods, where a solution pattern is formed and repeated iteratively. With these methods, only relatively few Pareto optimal solutions have to be generated and evaluated, and the decision maker can further adjust his/her preferences as the solution process continues. In contrast to several other methods, the decision maker does not need to have knowledge about the global preference structure before the solution process. Due to the interactive solution process he/she will learn about the nature of the problem and will most likely have more confidence in the final solution.

One example of methods of this type is NIMBUS [31, S. 5.12.], [33], [35]. It is an interactive classification based multiobjective optimization method designed especially for an efficient handling of nonlinear problems. For that reason, it is capable of solving complicated real-world problems. The NIMBUS method has been successfully applied to some real world engineering problems, such as chemical process design [17], paper machine design [28], ultrasonic transducer design [20], and internal combustion engine design [2].

Each interactive method must be started from some initial solution. In the case of NIMBUS, this solution is either a solution specified by the decision maker, or a so called neutral compromise solution [35]. For the initial Pareto optimal solution, the decision maker produces in NIMBUS a classification, i.e. decides whether each of the objective function values should be improved, could be degraded, or if it should keep the current value. By this preference information the decision maker indicates how the current solution should be improved. Then one or more Pareto optimal solutions are created, and the decision maker may select one of them as a basis for a new classification. With an interactive procedure the decision maker proceeds stepwise towards the final solution, whose objective function values represent the most satisfying compromise to him/her.

In the NIMBUS method, the interaction phase has been aimed at being comparatively simple and easy to understand for the decision maker. At each iteration, the NIMBUS method offers flexible ways to direct the search for the best Pareto optimal solution according to the decision maker's wishes by means of classification. Classification does not require consistency from the decision maker and reflects his/her wishes well. The use of classification avoids using difficult and artificial concepts for extracting preference information.

The classification of the objective functions means that the designer indicates what kinds of improvements are desirable and what kinds of impairments are tolerable in objective function values in order to get a better solution than the current one. The basic idea in classification is that the designer contemplates the current Pareto optimal objective function values $f_i(x^c)$ at each iteration of the NIMBUS method and assigns each of the objective functions f_i into one of the following five classes depending on his/her preferences:

- I^{imp}, function value should be improved as much as possible,
- I^{asp}, function value should be improved to a certain aspiration level,

- I^{sat}, function value is satisfactory at the moment,
- I^{bound}, function value is allowed to impair to a certain acceptable bound, and
- I^{free}, function value is temporarily allowed to change freely.

While classifying Pareto optimal solutions, it must be remembered that if the designer wishes to improve some objective(s), at the same time (s)he must accept impairment in some other objective(s). Based on the classification, we can form a reference point $\bar{z}_i$, which consists of desirable values for each of the objectives. In other words, components of the reference point are derived depending on in which class each objective function belongs to. With regard to the classes above, corresponding components of the reference point are selected as follows: $I^{imp} \rightarrow \bar{z}_i = z_i^*$, $I^{asp} \rightarrow \bar{z}_i$=aspiration level, $I^{sat} \rightarrow \bar{z}_i$=current value $f_i(x^c)$, $I^{bound} \rightarrow \bar{z}_i$=acceptable bound and $I^{free} \rightarrow \bar{z}_i = z_i^{nad}$.

In the synchronous version of the NIMBUS method [35], several scalarizing functions leading to different subproblems, may be utilized. This is motivated by the fact that different scalarizing functions based on the same preference information may produce different solutions [34]. The decision maker can choose to see one to four different Pareto optimal solutions. In this study, based on the only negligible differences between different scalarizations with this problem, we employed only one of the four scalarizing functions, namely the one based on an achievement scalarizing function [48], which is formulated as:

$$minimize \ \max_{i=1,\ldots,k} \left[\frac{\bar{z}_i - f_i(x)}{z_i^{**} - z_i^{nad}} \right] + \rho \sum_{i=1}^{k} \frac{f_i(x)}{z_i^{**} - z_i^{nad}}, \tag{2}$$

$$\text{subject to } x \in S.$$

where for each $i = 1, ..., k$
$\bar{z}_i$ = component of the reference point
z_i^{nad} = approximated nadir objective vector component
z_i^{**} = approximated utopian objective vector component
ρ = some small positive value.
It is quaranteed that solution of (2) is Pareto optimal [35].

Besides classification, with NIMBUS the designer can also generate an arbitrary number of intermediate solutions between any two Pareto optimal solutions found so far, and use them as a base for a new classification, if desired. When the decision maker finds a Pareto optimal solution that satisfies his/her preference information with conflicting objectives, this is regarded as the final solution.

With an interactive decision process the designer can learn about the nature and interdependencies of the problem. By directing the solution process with the preference information it is possible to learn how objectives affect each other (trade offs), and also what kinds of solutions in general can or cannot be reached. This gives the designer a good opportunity to solve the problem as a whole, instead of having to

consider each of the objectives at a time. A deeper understanding of the problem most likely also makes the designer feel more confident of the quality of the final solution.

4 Interactive method with increasing accuracy

Our approach arises from the use of an interactive multiobjective optimization method, and the nature of interactive solution processes. At the beginning of an interactive optimization process, the designer is many times also at the beginning of a learning curve: (s)he is learning of how different objectives are interrelated, what the trade-offs between the objectives are and, also, what kind of objective values can be reached in general. After the learning phase, the designer has a better understanding of the problem, and (s)he can decide what the most satisfactory objective function values for the final solution are.

In our approach, we wish to demonstrate that the computational cost can be affected by adjusting the required accuracy of the solutions of the optimization algorithm (by using a predetermined threshold for objective function evaluations in each iteration) during the interactive solution process. At the beginning of the process, while the designer is still learning about the problem, quite a coarse accuracy may be used, and only rather small amount of objective function evaluations is needed to get grasp of the problem. As the designer learns more about the problem and its behavior, and consequently feels more confident about it, (s)he may approach the final solution with an ever increasing accuracy and number of objective function evaluations. As the final step of the optimization procedure, the final solution can be calculated using the budget that assures the designer of a good enough solution quality.

One important issue with this kind of adjustable approach is to help the decision maker to judge when a sufficient amount of objective function evaluations at each iteration has been reached. Although several tools of different types could be developed, for example, based on the scalarizing function value, the solution quality or on the budget of allowed objective function evaluations, we want to provide the decision maker with an easy and intuitive way to monitor how the solution process is progressing.

To this end, we propose a measure, which we refer to as a maximum difference percentage (MDP), to assess the quality of each solution with regard to the given classification. As the name suggests, the maximum difference percentage is the maximum difference between the components of the current solution and the given reference point of classification as percentages for each of the objectives, and formally it is calculated as

$$MDP = \max_{i=1,\ldots,k} \left[\frac{\bar{z}_i - f_i(x)}{\bar{z}_i} \right].$$

This formulation bears some resemblance with the structure of the achievement scalarizing function presented in Subsection 3.2.

As we are discussing simulation based optimization, it may be useful to relate the sufficient amount of objective function evaluations to the the accuracy of the simulation software itself. With, for example, engine simulators accuracy above, say, 95 percent can generally be considered good, and thus it is useless to solve the problem itself beyond the inherent accuracy of the simulator. In the following, we employ this idea with regard to MDP values.

With the continuously plotted values of MDP the decision maker can extract at least three different types of information about the status of the current optimization run, and thus decide when a sufficient amount of objective function evaluations has been used.

Firstly, the decision maker can readily see as a percentage how far the worst objective/component of the current solution is from the given reference point. This information can be related to some extent to the accuracy of the simulator used. For example, in some sense, if it is known that the simulator error is around 5%, it may be sufficient to stop computation when the MDP first drops below 5%.

Anyhow, it should be noted that MDP is always related to the given reference point. As a result of that, if the reference point is impossible to reach, MDP will never reach satisfying values. Vice versa, if given reference point is very easy to reach, MDP values will evidently fall below zero.

Secondly, by observing how values of the MDP develop while number of objective function evaluations grows, the decision maker can see if there is a trend of improving values (curve falling downwards), or whether the improvement has already stagnated (curve almost horizontal). When the stagnation occurs, it is reasonable to assume that sufficient amount of objective function evaluations has been reached.

Thirdly, the decision maker can observe the variation of MDP values ("tail thickness"). Usually, in the beginning of the optimization run (using a population based global solver), the values are varying widely, but after certain amount of evaluations, the variation of values starts to diminish. This implies with population based optimization algorithms that the population is converging. Again, the amount of variation can be related to the accuracy of the simulator used. For example, if the simulator error is the same 5% as above, it may be reasonable to stop computation at the latest when variation in MDP values approaches the range of 5%.

There are several other distance measures (between the reference point and the current solution) that could have been utilized, but the MDP was developed and chosen because of its resemblance to the type of scalarizing function employed, and its intuitive meaning to the decision maker, as percentages are readily understandable, unlike scalarized objective function values. Further, as the MDP measure gives maximum difference of the solution components to the reference point, no component of the current solution can be worse than the MDP value indicates.

To sum up the main building blocks of our method, let us remind that as a general framework of the interactive optimization system of our study we have chosen to use the interactive NIMBUS method (discussed in Subsection 3.2). Multiple ob-

jectives are scalarized using the scalarizing function (2), and as global optimizers we use CRS and DE algorithms. Further, in every iteration of the NIMBUS method we employ the MDP values to judge a sufficient amount for objective function evaluations.

Logical steps of our method are as follows:

1. At the beginning of the procedure the initial solution for NIMBUS (e.g. neutral compromise solution) is calculated.

2. In a classification step, based on the current solution, the decision maker classifies objective functions and adjusts desired values for objectives to reflect his/her preference information. In this step, the decision maker can also decide to create intermediate solutions (instead of classifying) between any two already known Pareto optimal solutions.

3. The decision maker may define a stopping criterion for the next iteration, by giving a threshold values for MDP, or for the variation in MDP ("tail thickness") values. If this stopping criterion is not given, the decision maker can stop the optimization run based on the continuously plotted MDP graph whenever it seems reasonable.

4. If the decision maker wishes to further refine the solution either with a new classification or with a higher accuracy, he/she may continue the process from the Step 2.

In Section 6 we demonstrate our method, and show how much the amount of objective function evaluations could be decreased in our example problem without deteriorating the overall quality of the solution process.

5 Example case of IC engine design optimization

To elucidate our approach, in the following paragraphs we shortly introduce an example problem using the particular building blocks of a general system depicted in Subsection 2.1. The optimization system setup on a general level is similar to the ones presented in [1, 2], and further, we use the same engine design problem as in [2] to allow the comparison between proposed approaches in this and the previous paper. We study the performance of a two stroke engine and optimize it by altering the shape of the exhaust pipe, while we concentrate on improving the computational efficiency.

Objective functions

Our objective functions represent three different properties of the engine or engine/vehicle combination. Two of these properties are derivatives of the engine performance only, namely *maximum power* and *bmep* (efficiency). The third one, namely

coverage, reflects compatibility of the engine properties with gearing and transmission from the perspective of vehicle performance. These objectives are covered in detail in [1].

Maximum power is simply the maximum output of the engine. Besides the maximum power we are also interested in how that power is produced. There are only two ways to improve the maximum power of an engine: by increasing the operating speed of the engine or by increasing the average pressure inside the cylinder during the power stroke (volumetric efficiency). The latter is to be preferred, since increasing the operating speed of the engine results in increased mechanical stresses, decreased mechanical efficiency due to increased friction, and exposes the engine to a failure. As the measure of volumetric efficiency, we use the factor which describes the average pressure inside the cylinder during the power stroke and it is known as *bmep* (brake mean effective pressure).

In addition to maximum power and bmep, we are interested in *coverage*, which reflects how well some specific engine suits a particular gearing. In essence, coverage is the ratio between maximum performance of the engine if maximum power was usable at all speeds, compared to real performance which is degraded by the fact that due to different gear ratios, only some fraction of maximum power is generally usable.

In real-life, the overall performance of an engine/vehicle combination can be determined and controlled by three factors: coverage, maximum power and *bmep*. It is possible to have an engine-gearbox combination with a very high coverage, while the peak power of the engine is low. This is obviously not a good solution. Therefore, the coverage and the maximum power are conflicting objectives and they must be considered in unison. In addition, *bmep* must be taken into account in order to control rpm (revolutions per minute) for the maximum power. Thus, when considering the goodness of a particular vehicle as a whole (including engine-gearbox combination), all the three factors must be studied simultaneously as three objective functions.

Engine configuration modelling

In this study, engine configuration modelling refers to a dual technique for the exhaust pipe shape representation and it should not be mixed with the engine model in the simulator. On the one hand, an engine configuration model is used to produce a general and modifiable exhaust pipe shape using as few design variables as possible (to lower the dimension of the optimization problem, and thus improving the computational efficiency). On the other hand, the exhaust pipe shape which is created using as few design variables as possible must be converted to a form which is usable for the simulator (i.e., a list of lengths and diameters). For example, a diffuser shape can be represented by a continuous curve. For the simulator, this continuous shape must be converted to separate sections whose start diameter, end diameter, and length is given.

In this study we use a so-called Bezier model developed in [1] to represent the shape of an exhaust pipe. This model [2] is capable of representing very dissimilar

pipe shapes using a Bezier curve and only six design variables. The first of them is the total length of the pipe, which is defined similarly to the empirical models given in [8]. The next four variables are x and y coordinates for the two control points of the Bezier curve. The last design variable, the tail pipe diameter coefficient, determines the diameter of the end pipe, that is, the stinger. Bounds for the variables are given in [1],[2], and they define the feasible region S, that is, the search space.

The simulator

In this study, values for the three objective functions are derived from output files of an engine simulator. The simulator, MOTA 6.1 by J. van Leersum [27] and Ian Williams Tuning [49], was selected because it was necessary to consider especially the time consumed in a single simulation run and the possibility to link the simulator with an optimization tool. The execution time of MOTA for a single simulation run is a good average for an example case of simulation based optimization. A single run takes typically a few minutes using a modern PC (AMD Athlon 2.09 GHz, 1.0 GB of RAM), depending on the complexity of the pipe design and the grid granularity (total amount of rpm steps) for the end results.

MOTA is widely used among the two-stroke enthusiasts and semi-professional engine designers. The engine model used in MOTA is described in [27].

Optimization problem

Now, having defined our objective functions, we can formulate our multiobjective optimization problem

$$\begin{array}{ll} \text{maximize} & \{f_1(x), f_2(x), f_3(x)\} \\ \text{subject to} & x \in S, \end{array} \tag{3}$$

where we have six design variables $x = (x_1, \ldots, x_6)$, and $f_1(x)$ represents the maximum power of the engine, $f_2(x)$ describes the coverage and $f_3(x)$ stands for the effectiveness of the engine (bmep).

The feasible design variable space S is limited by upper and lower bounds for each variable and, thus, the problem has box constraints. Specifically, the tuned length L_t of the pipe was bounded to be between 700 and 800 mm.

6 Numerical example - solution process with NIMBUS with increasing accuracy

In this section, we describe the interactive solution process when a design problem of an internal combustion engine was solved with the IND-NIMBUS® software [32] (implementation of the NIMBUS method) and the CRS2 and DE algorithms were used as underlying global single objective solvers. We made three separate runs: two different runs using the CRS2 algorithm, and the second CRS2 run was duplicated using the DE algorithm (instead of CRS2) in order to see whether the optimization algorithm itself has major impact on the solution process.

For the first CRS run the MDP plots are omitted, and focus is merely on the numerical values. On the other hand, for the second CRS run, and corresponding DE run we present also MDP plots to show how they can be employed, and we wish that these plots may also give some insight to the behaviour of two different optimization algorithms. It is worth to mention, that in these runs MDP values were not actually used to stop the optimization procedure. Rather, each iteration was executed to the maximum number of evaluations, but we give some suggestions retrospectively how process could have been terminated based on MDP values and plots. Each of three runs is discussed more detail below in their own subsections.

For each run and each iteration of the interactive solution process we report the best objective function values gained at various levels of objective function evaluations. Number of objective function evaluations also corresponds to the number of simulator runs, since all three objective function values are derived from output files of the single simulator run.

As mentioned earlier, the synchronous NIMBUS algorithm can produce up to four different Pareto optimal solutions after each classification. In preliminary testing [1], [2] it turned out that in this example differences between them were quite small and, thus, only a single Pareto optimal solution was calculated in order to decrease the computational burden. This solution is produced by the so-called achievement scalarizing function, presented for example in [35],[48]. In the following, we show solutions where objective function values are in the order (power (in hp), coverage, bmep (in bar)). The objective function devoted to coverage has no units because it describes a ratio where the maximum value is 1.

Some information about the ranges of the objective functions in the Pareto optimal set is used for scaling purposes in NIMBUS. Normally, IND-NIMBUS® initially calculates so-called ideal (upper bound) and nadir (lower bound) objective vectors [31]. In order to save computation time, information for scaling was given here manually. The lower bounds were set as $(0.00, 0.00, 0.00)$, and the upper bounds as $(34.00, 1.00, 15.00)$. The values for both vectors were selected based on a decision maker's expertise, so that they are not reachable in real-life.

In Tables 1, 2 3, and 4, we present summaries of three different solution processes (first example is divided in to Tables 1 and 2 for space limitations) where a different budget for objective function evaluations were used at each iteration. The purpose of this is to see how the overall solution process is affected by the number of function evaluations used, and whether it is possible or not to reduce the evaluation count in the early phases of the optimization process without degrading the final solution. This design problem is similar to the one presented in [2] to allow the performance comparison, and the solution process follows quite closely classifications made in that study.

In rows of Tables 1, 2, 3, and 4 below horizontal lines we show the classifications made at each iteration with the desirable objective function values (Cls), and below them the corresponding values for each of the objectives at different evaluation levels. In order to see how the solution process is affected by the number of the objective function evaluations, at each iteration, results can be examined with

increments of 30 objective function evaluations, this number corresponding to the population size of CRS and DE algorithms. These increments of 30 evaluations are seen in "Eval level" column. Column "Best", gives the evaluation number where so far the best scalarized objective function value occurred before the given evaluation level. The next three columns display values for each of the separate objective functions (power, coverage, and bmep), or the given classification in the form of desired objective function values. For clarity, objective function values at evaluation levels where no further improvement was gained are left empty. In the last column "Scal. f Δ", the absolute change within each iteration in the scalarizing function value between the initial sampling phase and the best solution at that iteration is displayed.

It is necessary to mention that the form of the achievement scalarizing function [35, 48] we used here, does not require the information about the current solution. Thus, it was not necessary to replicate results of the whole run using solutions gained at different evaluation levels as the current solution for the next classification.

Some examples of how the MDP values can be used to judge sufficient number of objective function evaluations are discussed in Subsection 6.4.

6.1 First CRS run

In our first test run we were aiming at gaining good overall performance by having quite high maximum power, reasonable coverage, and fairly high efficiency. In Iteration 1, to begin the interactive solution process, the neutral compromise solution, which should be located roughly in the middle of the Pareto optimal set, was calculated as a starting point for the classification using 240 function evaluations. After that, the following iterations were executed also up to 240 function evaluations for each classification. The best objective function values are reported in the steps of 30 evaluations up to 240 function evaluations. The final iteration of the optimization procedure was executed allowing 480 function evaluations, in order to get a more accurate final result.

The neutral compromise solution for the problem was $(26.17, 0.79, 11.54)$. Actually quite a similar solution, $(26.02, 0.78, 11.25)$, was found already in the initial sampling phase (during 30 first random evaluations to create the initial population) of the CRS algorithm at the 28th evaluation. After that *bmep* and power improved slightly.

In Iteration 2, a classification was done in order to improve the neutral solution. The aim was to gain a higher coverage and for that reason power was allowed to decrease till 25.00, coverage was desired to improve till 0.85 and for *bmep*, a slight decrease till 11.00 was allowed. We can see that the solution $(24.88, 0.85, 10.98)$ pretty close to the classification specified could be found very early, also in the initial sampling phase as with the neutral compromise solution, and this solution was slightly improved with further evaluations to a final value $(25.13, 0.86, 11.09)$.

In Iteration 3, the classification was continued from the result of Iteration 2, and

Table 1: First example. Effect of the number of function evaluations to the interactive solution process.

Eval level	Best	Power (hp)	Cov	Bmep (bar)	Scal. f	Scal. f Δ
Iter 1	**Neut[a]**					
30	28	26.02	0.78	11.25	-0.2475	
60	28	-	-	-	-	
90	28	-	-	-	-	
120	118	26.30	0.77	11.38	-0.2393	
150	127	26.17	0.77	11.55	-0.2304	
180	169	26.17	0.79	11.54	-0.2281	
210	169	-	-	-	-	
240	169	-	-	-	-	0.0194
Iter 2	**Cls**	I^{bound}(25.00)	I^{asp}(0.85)	I^{bound}(11.00)		
30	5	24.88	0.85	10.98	0.0011	
60	5	-	-	-	-	
90	5	-	-	-	-	
120	116	25.09	0.85	11.07	0.0021	
150	116	-	-	-	-	
180	151	25.13	0.86	11.09	0.0062	
210	151	-	-	-	-	
240	151	-	-	-	-	0.0073
Iter 3	**Cls**	I^{bound}(24.00)	I^{asp}(0.92)	I^{asp}(11.50)		
30	23	23.08	0.87	10.83	0.0452	
60	23	-	-	-	-	
90	23	-	-	-	-	
120	104	22.77	0.88	10.92	0.0401	
150	104	-	-	-	-	
180	104	-	-	-	-	
210	104	-	-	-	-	
240	235	23.38	0.88	10.97	0.0347	0.0104
Iter 4	**Cls**	I^{bound}(22.00)	I^{asp}(0.90)	I^{sal}(10.80)		
30	22	23.08	0.87	10.83	-0.0252	
60	32	23.89	0.88	10.54	-0.0205	
90	32	-	-	-	-	
120	99	23.55	0.90	10.82	0.0004	
150	99	-	-	-	-	
180	162	22.60	0.91	10.84	0.0048	
210	162	-	-	-	-	
240	162	-	-	-	-	0.0300
Iter 5	**From**	22.60	0.91	10.80		
	To	25.00	0.85	11.00		
	1					
30	23	23.08	0.87	10.83	-0.0167	
60	23	-	-	-	-	
90	83	23.22	0.89	10.67	-0.0152	
120	111	22.91	0.89	10.75	-0.0099	
150	111	-	-	-	-	
180	111	-	-	-	-	
210	111	-	-	-	-	
240	240	23.00	0.89	10.80	-0.0070	0.0096
	2					
30	22	24.36	0.84	11.43	-0.0498	
60	54	23.79	0.85	10.71	-0.0356	
90	82	24.11	0.89	11.08	-0.0029	
120	82	-	-	-	-	
150	130	23.74	0.89	11.14	0.0015	
180	130	-	-	-	-	
210	130	-	-	-	-	
240	130	-	-	-	-	0.0512
	3					
30	1	24.59	0.85	10.85	-0.0230	
60	46	23.96	0.86	11.01	-0.0122	
90	65	24.59	0.86	11.30	-0.0047	
120	65	-	-	-	-	
150	128	24.24	0.88	11.14	-0.0037	
180	128	-	-	-	-	
210	128	-	-	-	-	
240	128	-	-	-	-	0.0193
	4					
30	1	23.44	0.83	10.55	-0.0372	
60	1	-	-	-	-	
90	87	24.41	0.86	11.22	-0.0123	
150	87	-	-	-	-	
180	87	-	-	-	-	
210	198	24.73	0.86	11.13	-0.0082	
240	217	24.67	0.87	11.10	-0.0022	0.0350

Table 2: First example, continued.

Eval level	Best	Power (hp)	Cov	Bmep (bar)	Scal. f	Scal. f Δ
Iter 6	Cls	$f^{asp}(25.50)$	$f^{asp}(0.88)$	$f^{asp}(11.20)$		
30	15	16.92	0.86	7.94	-0.0254	
60	15	-	-	-	-	
90	15	-	-	-	-	
120	87	-	-	-	-	
120	99	24.58	0.86	11.06	-0.0249	
150	137	25.08	0.87	11.07	-0.0099	
180	137	-	-	-	-	
210	137	-	-	-	-	
240	137	-	-	-	-	
270	137	-	-	-	-	
300	137	-	-	-	-	
330	137	-	-	-	-	
360	137	-	-	-	-	
390	137	-	-	-	-	
420	410	25.10	0.88	11.07	-0.0093	
450	437	25.16	0.87	11.10	-0.0076	
480	137	-	-	-	-	0.0179

the aim was to gain even more coverage and slightly more *bmep* and, thus, power was allowed to decrease till 24.00, and coverage and *bmep* were hoped to improve till 0.92 and 11.50, respectively. Even the final solution $(23.38, 0.88, 10.97)$ of this iteration gained with 235 evaluations failed to reach the given classification, suggesting that high coverage is preventive for power and *bmep*. Also in this case differences between solutions with different evaluation levels remained quite small.

In Iteration 4, we continued from the previous solution. In the quest for higher coverage till 0.90, power was allowed to decrease further till 22.00, and for the *bmep* a slight degradation till 10.80 was regarded acceptable. Solution $(22.60, 0.91, 10.84)$ was obtained at the 162nd evaluation, and at lower evaluation levels we see that continuous decrement of power levels was traded with higher coverage levels. In this iteration, solution $(23.08, 0.87, 10.83)$ gained at the CRS2 algorithm's initial sampling phase at the 22nd evaluation is improved quite continuously with an increasing number of evaluations, resulting also in one of the biggest improvements in scalarizing function value compared to all iterations within this test run.

At this stage of the solution process we wanted to study more Pareto optimal solutions to see how the behavior of power and coverage interacts. To achieve this, in Iteration 5 we used the option of generating new Pareto optimal alternatives between two already known Pareto optimal solutions. As the end points two already known solutions we used: $(25.13, 0.86, 11.09)$ from Iteration 2 (which has a good value for power) and $(22.60, 0.91, 10.84)$ from Iteration 4 (which has a good value for coverage). Between these points we chose to get 4 new solutions, which were $(23.00, 0.89, 10.80)$, $(23.74, 0.89, 11.14)$, $(23.24, 0.88, 11.14)$, and $(24.67, 0.87, 11.10)$. The last of these results, $(24.57, 0.87, 11.10)$, encouraged us to believe that there could exist a solution with a power level above 25 hp, with coverage close to 0.90, and *bmep* over 11.

In the first new alternative of the Iteration 5 the solution of initial sampling was only slightly improved with more evaluations, as is also seen in the improvement of the scalarizing function value. In the second new alternative the initial solution is obviously changing, but for the human observer it is not clear how much the solution actually improves. By the change of scalarizing function value, this solution has improved most with an increasing number of evaluations during this experiment.

The third new alternative does not seem to benefit much of increasing evaluation numbers as the solutions are quite close to each other. The fourth new alternative improves all the way through the increasing number of evaluations, and also the change in scalarizing function value is quite big.

Although the last intermediate solution of the Iteration 5 was rather satisfactory, we wanted to take one more iteration using more function evaluations in order to achieve a high accuracy for the final solution, reflecting our possibly optimistic beliefs. With our final classification we aimed at a quite high demand for power as we hoped it to improve till 25.50, similarly we hoped to improve *bmep* and coverage till 0.88 and 11.20, respectively. As this classification is not allowed by definition of Pareto optimality from the last result of the previous iteration (one cannot improve all the objectives at the same time), we used the neutral compromise solution as the base for the new classification. This procedure does not compromise integrity of the solution process, since our optimization algorithm is global instead of a local one, and as such it does not employ information about the current solution. Instead, the initial population of the CRS2 algorithm is always created randomly, regardless of the current solution. Besides that, contrary to some sclarizing functions [34] the scalarizing function used here does not employ information about the current solution.

With 480 function evaluations devoted for getting the solution in the final iteration, we obtained solution $(25.16, 0.87, 11.10)$. Although the desired levels were not quite achievable, this solution was considered satisfactory and selected as the final solution. In this iteration, with an increasing number of function evaluations, it seems difficult to gain any improvement, and although 480 evaluations were used, only minor change in scalarizing function value was achieved. This probably suggests that the given classification is close to the performance limits of the engine to be optimized. In our final solution, the power peak settled at a reasonable level with 25.16 hp at 12500 rpm and also coverage and *bmep* reached good values.

6.2 Second CRS run

Because it seemed that our example problem is not very sensitive to the number of objective function evaluations, we wanted to change our classifications, and pursue other kind of trade-off between our objectives. Thus, in the second example we were pursuing very high coverage with the expense of both bmep and power. A summary of results is seen in Table 3. Results of Iteration 1 are naturally the same as the ones in the first example because we begun with the same neutral compromise solution as a starting point.

In Iteration 2 we allowed power and bmep to decrease till 23.00 and 10.50, respectively, and at the same time we were aiming to get coverage at least to 0.94. The solution changed quite a lot during the evaluations of this iteration, as also the change in the scalarizing function value suggests. With the solution $(22.43, 0.92, 10.76)$, power fell below requirement, coverage did not achieve the given value, and in bmep there

Table 3: Second example. Effect of the number of function evaluations to the interactive solution process.

Eval level	Best	Power (hp)	Cov	Bmep (bar)	Scal. f	Scal. f Δ
Iter 1	**Neut***					
30	28	26.02	0.78	11.25	-0.2475	
60	28	"	"	"	"	
90	28	"	"	"	"	
120	118	26.30	0.77	11.38	-0.2393	
150	127	26.17	0.77	11.55	-0.2304	
180	169	26.17	0.79	11.54	-0.2281	
210	169	"	"	"	"	
240	169	"	"	"	"	0.0194
Iter 2	**Cls**	$I^{bound}(23.00)$	$I^{asp}(0.94)$	$I^{bound}(10.50)$		
30	23	23.08	0.87	10.83	-0.0651	
60	52	21.90	0.88	10.50	-0.0573	
90	69	21.53	0.91	10.32	-0.0410	
120	111	22.07	0.91	10.36	-0.0251	
150	111	"	"	"	"	
180	111	"	"	"	"	
210	197	22.33	0.92	10.48	-0.0175	
240	236	22.43	0.92	10.76	-0.0159	0.0492
Iter 3	**Cls**	$I^{bound}(22.00)$	$I^{asp}(0.94)$	$I^{bound}(10.00)$		
30	6	20.04	0.90	10.04	-0.0556	
60	37	21.31	0.93	10.22	-0.0180	
90	37	"	"	"	"	
120	115	21.47	0.92	10.08	-0.0174	
150	115	"	"	"	"	
180	176	21.92	0.93	10.07	-0.0096	
210	176	"	"	"	"	
240	211	21.81	0.93	10.02	-0.0070	0.0486
Iter 4	**Cls**	$I^{asp}(24.00)$	$I^{asp}(0.94)$	$I^{bound}(9.50)$		
30	23	23.08	0.87	10.83	-0.0651	
60	23	"	"	"	"	
90	82	22.42	0.88	10.75	-0.0595	
120	113	24.33	0.89	10.52	-0.0465	
150	144	23.54	0.91	10.38	-0.0300	
180	144	"	"	"	"	
210	207	23.64	0.91	10.23	-0.0259	
240	221	23.10	0.92	10.19	-0.0241	0.0411
Iter 5	**Cls**	$I^{bound}(21.00)$	$I^{asp}(0.98)$	$I^{bound}(9.00)$		
30	6	20.04	0.90	10.04	-0.0803	
60	6	"	"	"	"	
90	69	21.89	0.95	8.94	-0.0271	
120	69	"	"	"	"	
150	69	"	"	"	"	
180	69	"	"	"	"	
210	69	"	"	"	"	
240	69	"	"	"	"	0.0532

still was some room for acceptable decrement. Thus, it was not possible to obey the classification completely.

In Iteration 3 we were further compromising values of power and bmep by allowing them to decrease till 22.00 and 10.00, respectively. At the same time coverage was required to be at least 0.94. The solution $(20.04, 0.90, 10.04)$ of the initial sampling phase was improved quite a lot with additional evaluations. The final solution $(21.81, 0.93, 10.02)$ of this iteration could follow quite closely the given classification: power and coverage were only slightly below their minimum values, and bmep was little bit higher than required.

In Iteration 4, coverage requirement was still kept at 0.94, and power was required to reach higher for 24.00, and for bmep decrement further down to 9.50 was accepted. Also in this iteration the initial solution $(23.08, 0.87, 10.83)$ benefited from the additional evaluations, as the scalarizing function value changed quite a lot. Anyhow, with the the final solution $(23.10, 0.92, 10.19)$ of this iteration, power and coverage were lagging behind the desired ones, whereas bmep was essentially higher than required. Thus, it was not possible to find a feasible solution that would accurately correspond to the given classification.

In Iteration 5 we made the final attempt towards extremely high coverage by

setting a requirement for it to 0.98 (considering that 1.00 is an unreachable theoretical maximum this was a very optimistic setting), and further allowing power and bmep to decrease till 21.00 and 9.00, respectively. The solution of the initial sampling phase $(20.04, 0.90, 10.04)$ was improved only once at iteration 69 resulting with solution $(21.89, 0.95, 8.94)$. Regardless of the only single improved solution, the absolute improvement in the initial sampling and final scalarizing function value was greatest within the iterations of this example. As we were here aiming at the maximum coverage while having power and bmep values at least within some reasonable limits, we regarded solution $(21.89, 0.95, 8.94)$ of Iteration 5 as the final solution of this example. This high coverage with regard to reasonable values of maximum power and bmep may be close to a theoretical limit.

It is worth mentioning that direct comparison of scalarizing function values between different classifications (i.e. iterations) is not reasonable. This is because the scalarized problem to be solved, based on different classifications, is different. Thus, consideration of scalarization function values of different iterations with different classifications leads us to compare apples and oranges, so to speak. Anyhow, within one single iteration scalarized function values may give the decision maker some additional information about the solution development.

6.3 DE run

Our third run is identical to the second run, with regard to the classifications made. The only difference is that in this example we used DE as a solver, instead of CRS as in two previous examples. The results are seen in Table 4. The results of Iteration 1 are naturally the same as the ones in the two previous examples, because the initial (neutral compromise) solution is computed. In contrast to the two previous examples, Table 4 contains one additional column "Comp" to highlight performance differences between CRS and DE. Markings in this column indicate whether DE performed better (+), worse (-), or similarly (0) in that particular evaluation level compared to CRS in the previous example.

In iterations 2, 4 and 5 DE performed initially somewhat better, up to 90 evaluations, but beyond that, CRS performed exclusively better. As a consequence, total improvement in scalarized function value was higher with CRS in all iterations. For this reason, while the CRS method was used, the given classification was followed somewhat more accurately. Of course, these conclusions are not necessarily very well generalizeable beyond these examples.

Although the CRS performed slightly better in this example, differences in real objective function values were not very large, and thus it seems that our example problem is not very sensitive to the selection of the optimization algorithm. Thus, we can make some conclusions that are not limited merely to one global optimization method.

Table 4: Third example. Effect of the number of function evaluations to the interactive solution process. Similar to second example, but instead of the CRS, the DE algorithm was used.

Eval level	Best	Power (hp)	Cov	Bmep (bar)	Scal. f	Scal. f Δ	Comp
Iter 1	**Neut***						
30	28	26.02	0.78	11.25	-0.2475		0
60	28	-	-	-	-		0
90	28	-	-	-	-		0
120	118	26.30	0.77	11.38	-0.2393		0
150	127	26.17	0.77	11.55	-0.2304		0
180	169	26.17	0.79	11.54	-0.2281		0
210	169	-	-	-	-		0
240	169	-	-	-	-	0.0194	0
Iter 2	**Cls**	I^{bound}(23.00)	I^{asp}(0.94)	I^{bound}(10.50)			
30	23	23.08	0.87	10.83	-0.0651		0
60	47	21.92	0.90	10.29	-0.0378		+
90	47	-	-	-	-		+
120	117	22.23	0.90	10.43	-0.0344		-
150	117	-	-	-	-		-
180	117	-	-	-	-		-
210	181	22.38	0.91	10.50	-0.0321		-
240	181	-	-	-	-	0.0330	-
Iter 3	**Cls**	I^{bound}(22.00)	I^{asp}(0.94)	I^{bound}(10.00)			
30	6	20.04	0.90	10.04	-0.0556		0
60	36	21.24	0.89	9.97	-0.0455		-
90	88	22.11	0.89	9.96	-0.0445		-
120	115	21.99	0.92	9.90	-0.0216		-
150	115	-	-	-	-		-
180	115	-	-	-	-		-
210	195	22.15	0.92	9.97	-0.0139		-
240	219	21.93	0.93	10.08	-0.0095	0.0461	-
Iter 4	**Cls**	I^{asp}(24.00)	I^{asp}(0.94)	I^{bound}(9.50)			
30	23	23.08	0.87	10.83	-0.0651		0
60	47	21.92	0.90	10.29	-0.0590		+
90	47	-	-	-	-		+
120	47	-	-	-	-		-
150	135	23.09	0.88	9.79	-0.0550		-
180	135	-	-	-	-		-
210	205	22.05	0.92	9.73	-0.0550		-
240	237	22.39	0.89	10.51	-0.0519	0.0133	-
Iter 5	**Cls**	I^{bound}(21.00)	I^{asp}(0.98)	I^{bound}(9.00)			
30	6	20.04	0.90	10.04	-0.0803		0
60	32	19.40	0.93	9.95	-0.0512		+
90	32	-	-	-	-		-
120	102	19.40	0.95	9.30	-0.0450		-
150	102	-	-	-	-		-
180	102	-	-	-	-		-
210	102	-	-	-	-		-
240	226	21.87	0.94	9.85	-0.0408	0.0396	-

6.4 MDP plots

In Figure 1 we have collected eight MDP plots depicting two identical (with regard to classifications made) runs of Subsections 6.2 and 6.3. The first iteration, computation of neutral compromise solution, is omitted in both cases, because there is no preference information available, needed to compute the MDP value. In the figures, on x-axis is the number of function evaluations, and on y-axis is the MDP value of each objective function evaluation. In other words, any 30 adjacent points (with regard to x-axis) represent the current population of the optimization algorithm.

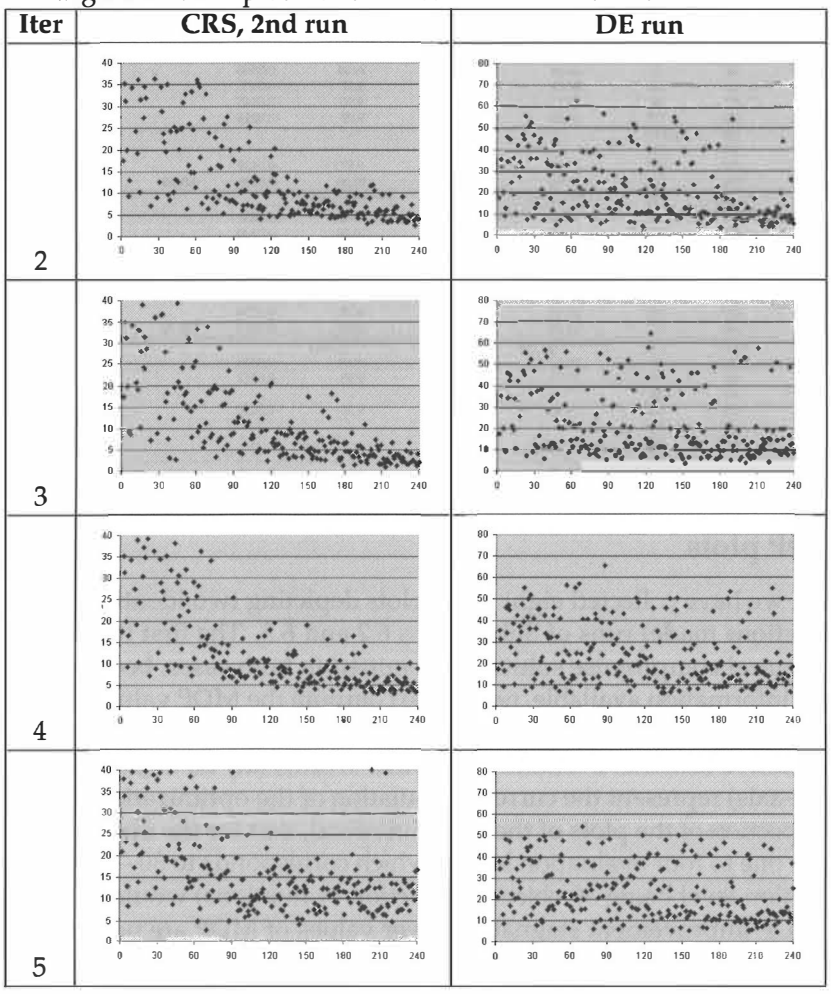
The inspection of the plots of the CRS run reveal, that for the majority of the iterations only one to four evaluation levels (which corresponds to 30 - 120 evaluations, one evaluation level being 30 objective function evaluations) would be sufficient to see that MDP improvement stagnates (lower values of MDP are not improving i.e. getting smaller), best MDP values are already quite small, and the variation in MDP has already reduced. Especially in the early iterations, while the decision maker is still learning about the problem, already one or two evaluation levels seem to show sufficient stagnation to stop the current iteration. This probably suggests that the

problem is quite easy to solve with regard to the the given classification.

In the plot of the 5th Iteration, it is seen that the MDP variance remains higher than in other plots. This is probably due to the more strict classification, i.e. the given classification is quite hard, or even impossible to reach.

The plots of the DE run are little bit more difficult to interpret, but the same behaviour as in the CRS run, is seen with somewhat higher variation (notice the different scaling of MDP values in y-axis). It is clearly visible in the variation of MDP within population, that the convergence rate of the CRS is somewhat faster (possibly due to the parameter settings of the DE), as also numerical results suggested.

Figure 1: MDP plots for the second CRS run and the DE run.



7 Discussion

In the examples reported, the interactive optimization procedure allowed us to guide the solution process to a desired direction step by step, and also to learn of what kind of solutions are attainable. In a majority of the iterations, only smallish number of 240 function evaluations were used but, yet, the solutions obtained followed the given classifications accurately enough to allow moving to a desired direction in the Pareto optimal set. In the final iteration of the first example, up to 480 function evaluations were used, and this lead to a sufficiently good final solution. The solution process was computationally efficient in all because in this example, only nine or five Pareto optimal solutions had to be generated to find the most preferred one. The whole optimization procedure consumed in the first run 2400 and in the second and third run 1200 objective function evaluations, which equals roughly to 48 and 24 hours of computation on a modern workstation.

While inspecting the data in all three tables and in MDP plots, we can deduce that only one to four evaluation levels (corresponding to 30 - 120 evaluations) for the majority of the iterations would be sufficient to be able to reconstruct a similar path of classifications as was done in the three examples. This suggests that the number of 1200 or 2400 evaluations could be cut down to approximately one third (400 or 800, respectively) without deteriorating the quality of the final solution using this technique. This would correspond roughly to 8 or 16 hours of computation, respectively. In other words, we could achieve considerable savings.

One major issue (also with general applicability) for further study for this kind of a technique is to develop more sophisticated tools to judge when a sufficient amount of objective function evaluations at each iteration has been reached. In our study we used a simple, but intuitive MDP plots, possibly with predefined thresholds. This idea could be further refined and automated by calculating more key figures of the plots, for example, median of MDP values with a sliding window could be plotted, as well as range wherein, for example, 80% of solutions reside. These mechanisms, especially with arbitrarily selected problems, remain topics for further study.

All the information above could be presented in real time to the decision maker as various graphical plots to help his/her judgement. Thus, either the decision maker or a heuristic based on MDP values could restrict the number of function evaluations to a sufficient level at each iteration. As one iteration may take several hours, the use of a heuristic is needed whenever the decision maker is not able to occasionally check the status of the optimization run, for example, during night time.

Concerning the classification, especially in the second example we noticed that while using only small amounts of objective function evaluations, it may be beneficial to set preferred function levels on to a bit optimistic side, as solutions satisfying the given classifications can not always be found. For example, in our second example maximum coverage was gained when requirement for it was set to a very high level. Anyhow, it is worth to mention that naturally not all classifications can be followed if they are too demanding, and this is also something that the decision maker must learn about the problem during the process.

While inspecting all three objective function values at various numbers of evaluations, we noticed that in several cases already sampling of the search space with a tiny initial population ($5 * d$, 30 in our case) of the CRS algorithm produced quite a good accuracy with regard to the given classification. On the other hand, especially in the second example, solutions continuously benefited from additional evaluations. In all three test runs, the last classification seemed to be hard to reach. This is probably due the accumulation of understanding of the problem behavior, and as a consequence of strive to achieve maximum performance, classifications in all cases were quite demanding. As a result of the strict classification the satisfactory solution is not so easy to find, indicating either a need for more objective function evaluations, or a less demanding classification. Either way, in the final stage of the optimization procedure, it may be reasonable to use higher number of objective function evaluations to be assured of a final high quality solution.

In the implementation of CRS, the initial sampling of the search space is done similarly at each iteration of the process, and it does not depend, for example, on the classification the decision maker made, or the current solution. For this reason, the first $5 * d$, (30 in our case) iterations in all iterations (except in the Iteration 5 of the first example where alternative solutions were created) are identical, and in this sense this is time wasted duplicating already known results. This behavior could be optionally changed by using a different random seed for the initial sampling in CRS2 algorithm, but it remains open to discussion whether this would benefit the overall process or not.

8 Conclusions

Our study arises from the fact that real-life engineering problems have very often multiple objectives, objective functions involved are highly nonlinear and they contain multiple local minima, and function values are often produced via a time-consuming simulation process. In this study, our aim was to show that the computational complexity (due to time consuming objective function evaluations) of interactive solution processes can be decreased. Our idea is to improve the efficiency of the whole interactive optimization system by exploiting fewer objective function evaluations in the beginning of the optimization procedure, by expense of the solution accuracy, and stopping each of the iterations when sufficient number of objective function evaluations has been reached. To help judge what is a sufficient number of objective function evaluations, we introduced a maximum difference percentage (MDP) measure.

Our example case concerned internal combustion engine design. More specifically, we optimized the performance of a two-stroke engine by altering the exhaust pipe shape. We formulated a multiobjective optimization problem using three objective functions measuring the goodness of a particular engine design. The problem was solved by the interactive NIMBUS method together with efficient global optimization algorithms (CRS2, DE), observing along the procedure how objective function values develop at growing levels of objective function evaluations.

As we noticed, in the early phases of the solution process it is not necessary to have very accurate solutions, but instead use more coarse accuracy to get grasp of the problem. While employing this principle, interactive methods may be a convenient way to reduce the number of objective function evaluations required, and yet be able to control the solution process. With this approach we could reduce the number of required objective function evaluations by some 60-70%, without deteriorating the overall solution process.

As the approach of this study was based on the concept of accuracy, one thing to consider in the field of simulation based optimization is the accuracy of the simulation software itself. With engine simulators accuracy above, say, 95 percent can generally be considered good, and thus it is useless to solve the problem itself beyond the inherent accuracy of the simulator. This further justifies our approach. Moreover, at least in our case, although the problem contains lots of local optima, equally good optima are rather easy to find, as results of this study also suggest.

With regard to the solution quality, we may notice that results gained by the optimization system of this study look very credible by the traditions of the trade. The optimized pipe shape resembles closely the ones seen in the aftermarket performance exhaust pipes, and respective power curve suggests high usability for the optimized pipe.

References

[1] T. Aittokoski (2007): On optimization of simulation based design. Licentiate Thesis. Jyväskylä Licentiate Thesis in Computing 8. University of Jyväskylä.

[2] T. Aittokoski and K. Miettinen (2007): Cost Effective Simulation-Based Multiobjective Optimization in Performance of Internal Combustion Engine. Engineering Optimization 40(7), 593-612.

[3] T. Aittokoski and K. Miettinen (2008): Efficient evolutionary method to approximate the Pareto optimal set in multiobjective optimization. In Proceedings of International Conference on Engineering Optimization EngOpt 2008, Rio de Janeiro, Brazil, June 1-5, 2008.

[4] M.M. Ali, C. Khompatraporn and Z.B. Zabinsky (2005): A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. Journal of Global Optimization 31, 635-672.

[5] M.M. Ali and C. Storey (1994): Modified Controlled Random Search Algorithms. International Journal of Computer Mathematics 54, 229-235.

[6] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty (1993): Nonlinear Programming: Theory and Algorithms. John Wiley and Sons, New York, 2nd edition.

[7] L.T. Biegler (1989): Chemical Process Simulation. Chemical Engineering Progress 85(10), 50-61.

[8] G. P. Blair (1996): Design and Simulation of Two-Stroke Engines. Society of Automotive Engineers, Inc. Warrendale, Pa.

[9] K. Deb (2001): Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester.

[10] K. Deb (0000). Unveiling Innovative Design Principles by Means of Multiple Confilicting Objectives. KanGAL Report Number 2002007.

[11] J. F. Elder IV (1992): Global Rd Optimization when Probes are Expensive: the GROPE Algorithm. Proceedings IEEE International Conference on Systems, Man, and Cybernetics, Chicago, Illinois, October 18-21, 1992.

[12] H. Eschenauer, J. Koski and A. Osyczka (editors) (1990): Multicriteria Design Optimization – Procedures and Applications. Springer-Verlag, Berlin.

[13] Y. Fu and U. M. Diwekar (2004): An Efficient Sampling Approach to Multiobjective Optimization. Annals of Operations Research 132, 109-134.

[14] A. Gaspar-Cunha and A. S. Vieira (2004):. A hybrid multi-objective evolutionary algorithm using an inverse neural network. Hybrid Metaheuristics (HM 2004) Workshop at ECAI 2004, 25-30.

[15] A. Gaspar-Cunha and A. Vieira (2005): A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. International Journal of Computers, Systems, and Signals 6(1), 18-36.

[16] P. E. Gill, W. Murray and M. H. Wright (1981): Practical Optimization. Academic Press, New York.

[17] J. Hakanen, K. Miettinen, M.M. Mäkelä and J. Manninen (2005): On Interactive Multiobjective optimization with NIMBUS in Chemical Process Design. Journal of Multi-Criteria Decision Analysis 13, 125-134.

[18] A-R. Hedar and M. Fukushima (2003): Minimizing multimodal functions by simplex coding genetic algorithm. Optimization Methods and Software 18, 265-282.

[19] A-R. Hedar and M. Fukushima (2003): Heuristic Pattern Search and Its Hybridization with Simulated Annealing for Nonlinear Global Optimization. Optimization Methods and Software 19, 291-308.

[20] E. Heikkola, K. Miettinen and P. Nieminen (2006): Multiobjective Optimization of an Ultrasonic Transducer using NIMBUS. Ultrasonics 44(4), 368-380.

[21] R. Horst and P. M. Pardalos (Eds.) (1995): Handbook of Global Optimization. Kluwer Academic Publishers, Boston.

[22] R. Horst, P.M. Pardalos and N.V. Thoai (2000): Introduction to Global Optimization, 2nd Edition. Kluwer Academic Publishers, Boston.

[23] W. Huyer and A. Neumaier (1999), Global optimization by multilevel coordinate search. Journal of Global Optimization 14, 331-355

[24] D.R. Jones, M.Schonlau and W.J.Welch (1998): Efficient Global Optimization of Expensive Black-Box Functions. Kluwer Academic Publishers, Boston.

[25] J. Knowles (2006): ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems. IEEE Transactions on Evolutionary Computation 10(1), 50-66.

[26] J. Knowles and E. J. Hughes (2005): Multiobjective Optimization on a Budget of 250 Evaluations. C. A. Coello Coello et al. (Eds.), Evolutionary Multi-Criterion Optimization 2005, Lecture Notes in Computer Science 3410, 176190. Springer-Verlag, Berlin.

[27] J. van Leersum (1998): A Numerical Model of A High Performance Two-Stroke Engine. Applied Numerical Mathematics 27, 83-108.

[28] E. Madetoja, K. Miettinen, P. Tarvainen (2006): Issues Related to the Computer Realization of a Multidisciplinary and Multiobjective Optimization System. Engineering with Computers 22(1), 33-46.

[29] R. T. Marler (2005): A Study of Multi-Objective Optimization Methods for Engineering Applications. A doctoral thesis in Mechanical Engineering in The Graduate College of The University of Iowa. http://www.digital-humans.org/Tim-Thesis.pdf.

[30] R. T. Marler and J. S. Arora (2004): Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26, 369-395.

[31] K. Miettinen (1999): Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston.

[32] K. Miettinen (2006): IND-NIMBUS for Demanding Interactive Multiobjective Optimization, in "Multiple Criteria Decision Making '05", Ed. by T. Trzaskalik, The Karol Adamiecki University of Economics in Katowice, Katowice, 137-150, 2006.

[33] K. Miettinen and M.M. Mäkelä (1995): Interactive Bundle-Based Method for Nondifferentiable Multiobjective Optimization: NIMBUS. Optimization 34, 231-246.

[34] K. Miettinen and M.M. Mäkelä (2002): On Scalarizing Functions in Multiobjective Optimization. OR Spectrum 24, 193-213.

[35] K. Miettinen and M.M. Mäkelä (2006): Synchronous Approach in Interactive Multiobjective Optimization. European Journal of Operational Research 170, 909-922.

[36] P. K. S. Nain and K. Deb (2002): A computationally effective multi-objective search and optimization technique using coarse-to-fine grain modeling. Technical Report Kangal Report No. 2002005, IITK, Kanpur, India, 2002.

[37] J.A. Nelder and R. Mead (1965): A Simplex Method for Function Minimization. Computer Journal 7, 308-313.

[38] P. M. Pardalos and H. E. Romeijn (Eds.) (2002): Handbook of Global Optimization, Volume 2. Kluwer Academic Publishers, Boston.

[39] C.D. Perttunen, D.R. Jones and B.E. Stuckman (1993): Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Application 79, 157-181.

[40] W.L. Price (1977): Global Optimization by Controlled Random Search. Computer Journal 20, 367-370.

[41] S. Ruzika and M. M. Wiecek (2005): Survey Paper - Approximation Methods in Multiobjective Programming. Journal of Optimization Theory and Applications 126(3), 473-501.

[42] M.J. Sasena (2002): Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations. Dissertation. University of Michigan.

[43] S. Shan and G. G. Wang (2005): An Efficient Pareto Set Identification Approach for Multiobjective Optimization on Black-Box Functions. Journal of Mechanical Design 127(5), 866-874.

[44] D.P. Solomatine (1998): Genetic and other global optimization algorithms - comparison and use in calibration problems. Balkema Publishers.

[45] W. Stadler and J. Dauer (1993): Multicriteria Optimization in Engineering: A Tutorial and Survey. Structural Optimization: Status and Promise. M. P. Kamat, ed., AIAA: Washington, D.C., 209-249.

[46] R. Storn, K. Price (1997): Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341-359.

[47] A. Törn and A. Zilinskas (1989): Global Optimization. Springer-Verlag, Berlin.

[48] A. P. Wierzbicki (1999): Reference point approaches. Published in Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications. T. Gal, T. J. Stewart and T. Hanne (editors). Kluwer Academic Publishers, Boston.

[49] I. Williams (referenced at 3.4.2007): Ian Williams Tuning Homepage. http://www.iwt.com.au/index2.html.

[50] B. Wilson, D. Cappellari, T. W. Simpson and M. Frecker (2001): Efficient Pareto Frontier Exploration using Surrogate Approximations. Optimization and Engineering 2, 31-50.

[51] T. Ye and S. Kalyanaraman (2003): A Recursive Random Search Algorithm For Large-Scale Network Parameter Configuration. Published in Proceedings of the 2003 ACM SIGMETRICS, 196-205.

[52] J.B. Vosa, A. Rizzib, D. Darracqc and E.H. Hirschel (2002): Navier-Stokes Solvers in European Aircraft Design. Progress in Aerospace Sciences 38, 601-697.

[53] E. Zitzler, K. Deb and L. Thiele (2000): Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173-195.