HYBRID APPROACH TO DESIGN OPTIMISATION: PRESERVE ACCURACY, REDUCE DIMENSIONALITY

Mariusz KAMOLA*†

*NASK, ul. Wawozowa 18, 02-796 Warsaw, Poland,

Mariusz.Kamola@nask.pl, fax: +48 22 3808 201

†Inst. of Control and Computation Engineering, Warsaw Univ. of Technology,

ul. Nowowiejska 15/19, 00-665 Warsaw, Poland, mkamola@ia.pw.edu.pl

The paper proposes a design procedure for creation of a robust and effective hybrid algorithm, tailored to and capable of carrying out a given design optimisation task. In the course of algorithm creation it is proposed to choose a small set of simple optimisation methods, out of which those best performing will constitute the hybrid algorithm. Method simplicity allows for implementing ad-hoc modifications if unexpected adverse features of the optimisation problem are found. It is postulated to model a system that is smaller but conceptually equivalent, whose model is much simpler than the original one — and can be used freely during the algorithm construction. Successful operation of the proposed approach is presented in two case studies (power plant set-point optimisation and waveguide bend shape optimisation). The proposed methodology is intended to be used by those having not much knowledge of the system or modelling technology, but having basic practice in optimisation. It is designed as a compromise between brute force optimisation and the design optimisation preceded by a refined study of the underlying problem. Special attention is paid to cases where simulation failures (regardless of their nature) form big obstacles in the course of optimisation process.

Keywords: design optimisation, simulation optimisation, surrogate model, analysis failure

1 Problem description

Let us consider optimisation problems where the scalar objective function value cannot be calculated otherwise than by some complex algorithm. Here, we call an algorithm complex when its overall operation cannot be represented by analytic formulas. It means the algorithm output is unpredictable.

The reasons for such complexity may be various. The first one is computation accuracy which may make even simple procedures (like square root computation) "complex" if the data representation precision is inadequate. However, sheer increasing of precision may not help if the algorithm itself is extremely sensitive (e.g. fractal computation).

In the above examples, the algorithm output was unpredictable but at least deterministic. This is not a strict requirement; in fact, in huge number of cases the algorithm introduces (fake) randomness reflecting the (true) nature of some real object — the randomness the optimisation procedure has to deal with.

Naturally, considered algorithms model behaviour of various systems; they are the mathematical models of those systems in different phases of their life. So, one can model a system still being under design, one can investigate the best way an existent object may operate, or one may try to track back the chain of events that have led to system malfunction or destruction. Probably this is the cause of so many names given to our optimisation problems. A common one is "design optimisation", another one is "computer aided engineering", yet another one is "computationally complex engineering", to end with "simulation-optimisation". All refer to optimisation problems whose features have been once described accurately by Sandia Laboratories (Sandia, 2006):

The coupling of optimization with complex computational methods is difficult, and optimization algorithms often fail to converge efficiently, if at all. The difficulties arise from the following traits, shared by many computational methods:

1. The time required to complete a single function evaluation with one parameter is large [...].
2. Analytic derivatives (w.r.t. the parameters) of the objective and constraint functions are frequently unavailable [...].
3. The parameters may be either continuous or discrete, or a combination of the two.
4. The objective or constraint functions may not be smooth or well-behaved; i.e. the response surfaces can be severely nonlinear, discontinuous, or even undefined in some regions of the parameter space. The existence of several local extrema (multi-modality) is common.
5. Convergence tolerances in embedded iteration schemes introduce nonsmoothness (noise) in the function evaluation response surface, which can result in inaccurate numerical gradients.
6. Each function evaluation may require an "initial guess". Function evaluation dependence on the initial guess can cause additional nonsmoothness in the response surface. Moreover, a solution may not be attainable for an inadequate guess, which can restrict the size of the allowable parameter changes.

Such problems impose very tough, if not to say contradictory, requirements on the optimisation routines, even if the dimensionality is moderate. On one hand, unpleasant properties of objective and constraints tempt to use brute force, i.e. some very unrefined routine, often executed massively in parallel. On the other, every function evaluation requires lengthy computations to be run. This implies that the selected optimisation routine should be efficient, effective and robust. Methodology of such routine construction is the key problem for everyone interested in design optimisation. This paper presents an approach for making an efficient, effective and robust optimisation routine that differs from widely known practices.

In addition to the above problem features, two important things must be said. One is the fact that it may be not known whether the unpleasant model properties are just the result of inaccuracy or are inherent to the modelled system. In plain words, there may be no "knob" for adjusting modelling accuracy — and the optimisation routine must

take the numerical model as is, without asking about the nature of its behaviour. This is related to the other feature: the numerical algorithm for system modelling is closed in a separate computation module. This piece of software may be completely opaque, with no possibility of inspecting its internal values (not to say about making any changes). It is frequently described as "black box" — with the difference that a true black box never explodes, while a third-party simulator — sometimes does.[1]

1.1 Problem formulation

Design optimisation problem can be defined formally as follows:

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) \ , \tag{1}$$

where $\mathbf{x}$ is a vector of decision variables, i.e. values passed as input to the numerical algorithm. The algorithm computes a vector of its output values $\mathbf{y}$, called dependent variables. Only then the performance index $f(\cdot)$ value can be computed. The operation of the numerical algorithm can be described formally as finding, for any given $\mathbf{x}$, such $\mathbf{y}$ that

$$\begin{cases} h_1(\mathbf{x}, \mathbf{y}) &= 0 \\ \vdots & \\ h_N(\mathbf{x}, \mathbf{y}) &= 0 \end{cases} \text{, in short, } \quad \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \ , \tag{2}$$

is satisfied. Here, $\mathbf{h}(\mathbf{x}, \mathbf{y})$ represents ties between input and output of the modelled system. The optimisation process is, usually, additionally constrained by

$$\text{a) } \ \mathbf{x} \in D_{\mathbf{x}} \ , \qquad\qquad \text{b) } \ \mathbf{y} \in D_{\mathbf{y}} \ . \tag{3}$$

[1] Making such analogy is completely in place here: possible malicious behaviour of simulators include hanging or casting an exception that may destroy the controlling optimisation module.

The domains $D_{\mathbf{x}}$ and $D_{\mathbf{y}}$ almost always have positive measure, and are well defined by regular inequality constraints. Their shape is not a factor complicating optimisation in any meaningful way, so in this paper they are assumed to be hypercubes, which is in fact a common case (Papalambros, 1988, pp. 383, 387).

Therefore, the set of feasible decision variable values is constrained threefold: explicitly by (3a), implicitly by the existence of solution to (2), and implicitly by (3b). The second constraint is particularly troublesome because, when violated, no output is calculated at all and no faintest idea is given what was the reason or degree of constraint violation.

Typical approach to solve design optimisation problems will be shortly presented in Section 2. Next, two practical problems of the considered kind will be presented in Section 3, and followed by the author's proposal of methodology for solving them that is alternative to the typical approach. Results of applying such methodology to the two example problems will be discussed in Section 4.

2 Standard methodology for design optimisation

Every calculation of the objective function can be perceived as a costly experiment consisting in performing an analysis of system behaviour. The analysis is usually done by simulation or modelling module, but it also may be done in real system. Such experiments are costly both in the sense of money and time consumed. The time is particularly inconvenient as it is an inevitable cost that cannot be substituted by anything else.

The widely recognised and applied strategy to overcome such difficulty is to perform optimisation using a "surrogate model", i.e. some cost-effective but less accurate model to calculate the objective value. All the rest, i.e. the modelling and optimisation techniques are just the effect of following the paradigm of simplified modelling. Let us discuss the three component strategies: how to choose points for surrogate model tuning, how to make such a model, and how to find optimum using such model.

2.1 Design of experiments

Selecting points for surrogate model tuning is called the design of experiments (DOE). The term comes from statistics and encompasses also techniques for detecting which decision variables have bigger, and which lesser impact on system performance. DOE is usually performed in the initial stage of design optimisation with purpose to detect the importance of decision variables, but it is also used to set up (and sometimes, update continuously) the surrogate model. The most popular strategies are: random sampling, factorial DOE and orthogonal arrays. In case of random or quasi-random sampling simulations are run for a number of feasible points with equal probability for each point of being selected. In factorial DOE the objective value is calculated for every combination of selected values of decision variables (in particular, those values can be the upper and lower decision variable bounds). Orthogonal arrays strategies, like Fisher's or Taguchi's, have the advantage of evaluating each decision variable importance even in presence of disturbances. Taguchi's strategy has become particularly popular in industrial design problems.

2.2 Preparation of a surrogate model

A number of techniques exist for construction of a surrogate or approximate model. Using such model responses for optimisation, instead of running the "true" experiments, is called response surface methodology (RSM). The term RSM nowadays means the application of simplified models, of which polynomial, neural, Bayesian and Krigging are the most popular. Originally, RSM was fitted by regression linear, quadratic, cubic (and so on) models, as more and more simulation output values were available with the optimisation progress. Also, neural networks can be used for approximate modelling as they can model functions of any complexity — however, one has to plan the network structure reasonably. Bayesian, numerically laborious, technique has the advantage of modelling distributions rather than crisp objective values, and Krigging method is considered to be a good compromise

between classical RSM and Bayesian regression.

Many design optimisation packages make it possible for user to specify a custom model template: using a model matching well a particular problem is worthy more than any general purpose standard RSM technique. Such problem-specific surrogate model can be written in some programming (typically, scripting) language, but it may be created with any other effective technique. Sometimes it is made of yet another coarse-grain simulator running orders of magnitude faster than the accurate one — cf. eg. (Plambeck *et al.*, 1996). It should be emphasised that in any stage of modelling and optimisation the dimensionality of the analysed problems remains unchanged (unlike in the methodology proposed further in this paper).

2.3 Application of optimisation methods

The type of optimisation methods used to solve design optimisation problems depends heavily on whether gradients of the objective function and constraints w.r.t. decision variables can be estimated and relied upon. If yes, successive quadratic programming (SQP) (Bazaraa *et al.*, 1993) and generalised reduced gradient (GRG) (Edgar and Himmelblau, 1988) methods are nowadays considered the most advanced and efficient techniques — but of course simpler first or second-order methods are also in use (Poloni *et al.*, 2005). If the gradients are not available, direct search methods are preferred: Nelder-Mead simplex search, simulated annealing, controlled simplex and complex searches of many kinds and, most of all, evolutionary strategies. Inherent inefficiency of direct search routines, i.e. the number of required objective function evaluations being often prohibitively large, can be mitigated to some extent with parallel processing (the most recently published solutions suggest to employ grid technologies for parallel optimisation).

Recapitulating, design optimisation consists of the interplay of three general components: DOE, RSM and optimisation — but the details of those interactions depend

strongly on the type of the problem and are often left to be defined by the user. Particularly, the way a surrogate model is used and fitted, can be different: it can be global (for the whole domain) or local, it can be made more accurate by adding more points or increasing original modelling accuracy, it can model all output variables or only selected and adequate ones. This is strongly linked with the type of optimisation routine, which may change with the progress of optimisation.

2.4 Present design optimisation packages

Let us perform a short overview of commercial integrated design optimisation tools present on the market. The goal is to show that issues of interest in this paper — support for simulation failures and flexibility in combining DOE, RSM and optimisation — are addressed with various attention. The questions are:

- Can the modelling be done by external "black-box" module, or must it use a proprietary modelling language?
- Can the optimisation be done by user-supplied routine rather than by a built-in one?
- Can the tool apply optimisation procedures autonomously and adaptively, according to detected problem features?
- Is RSM with user-supplied model supported?
- Are simulation failures safely handled?

The answers, collected from (Hyperworks, 2006; Eldred *et al.*, 2005; Scott, 2001; Synaps, 2003), have been put in Table 1. It can be noticed that, while the attention and freedom is given to simulation side (support for user supplied model and handling of failures), the choice of optimisation strategy can be controlled by the user in quite a limited way. The reason for such attitude is undoubtedly that the coupling of simulation with optimisation is

Package name	"black-box" model supported	opt. routine supplied by user	opt. method chosen adaptively	RSM model supplied by user	support for simulation failures
HyperWorks	no[1]	no[2]	?	?	yes[3]
DAKOTA	yes	no[4]	yes[5]	no	yes
ModelCenter	yes	yes	?	?	?
iSight	yes	no	?	yes	yes
Epogy	yes	yes	yes	yes	yes

[1]Specialised models provided for metal forming, extrusion, mechanic part motion etc.
[2]Interfaces to many solvers.
[3]Special treatment of singularities in simulations.
[4]Source code available — any changes possible.
[5]Sequence of standard methods can be determined in advance ordered by the user.

Table 1: Key features of selected design optimisation packages. The question mark means that relevant information was not easily attainable; it should be understood as "probably no".

the piece of work that *must* be done, and a relatively simple (though sometimes laborious) one. On the other hand, playing with optimisation settings is not obligatory, not to mention it requires a far better expertise.

2.5 Stochastic design

Stochastic design problems can be — and in fact are — treated like the deterministic ones, save for the changes to objective formulation. It is assumed that random inputs affect the model output, which may be therefore treated as a vector of random variables with some standard deviation. Consequently, the objective and constraints (3b) are redefined taking into account their randomness. One type of stochastic design problems is particularly popular — the N-σ design in which the solution is required to have a given confidence interval w.r.t. constraints, determined by N.

The key issue is how to infer about model output distribution for a given design. This is typically done by producing a number of realisations of random inputs values in a more or less systematic way. Then a number of simulations is done to find out how the cumulative distribution functions for model outputs might look like. Optimisation algorithms applied to such problems are deterministic. They work well if the number of random samples taken from inputs is adequate to optimisation accuracy.

3 Example problems and proposed methodology

The main innovation to design optimisation proposed in this paper consists in an alternative way of constructing the optimisation algorithm. The suggested procedure comes as the result of the author's experience in design optimisation of two practical problems: set-point optimisation for a power plant and shape optimisation for a microwave guide.

3.1 Power plant set-point optimisation

The problem of power systems modelling has been for long present in the community of power engineers, also the Polish ones (Portacha, 1969). As the modelled systems consist of many elements, modelling them using polynomial models usually fails due to lack of appropriate amount or sort of data that could be used for tuning them. So, another approach has been taken that utilises knowledge of experts, which is expressed in the form of a large set of equations describing phenomena taking place in power plant elements. Such a physical model spans diverse branches of science and is extremely sophisticated but allows to model system behaviour already at the stage of system design.

The structure of the considered coal industrial power plant is comprised of boilers, turbines and regeneration blocks, with water circulating through them in cycles; see App. A for the detailed diagram of the plant. The basic physical model comes from (Jankowski *et al.*, 1972); it is essentially in the form of (2) with some elements of $\mathbf{y}$ being hidden inside the simulation module. The equation of type (2) may be solved by finding, for a given $\mathbf{x}$,

$$\arg\min_{\mathbf{y}} \sum_{i=1}^{N} h_i^2(\mathbf{x}, \mathbf{y}) \tag{4}$$

by an optimisation procedure. Unfortunately, the solution of (4) satisfying (2) may not always exist. Such situations usually correspond to real-life cases where the plant reaches dangerous states and must be switched off. However, solving (2) with (4) leaves no chance

of tracing the reason for such emergency situations.

Another approach to solve (2) is to decompose it into subsystems (usually corresponding to plant functional blocks) that may be solved by calculation of unknown variables by means of consecutive substitutions. Violation of any equation in the set is cancelled out by iterative adjustments of selected elements of $\mathbf{y}$. (Those elements play a role analogous to that which whole $\mathbf{y}$ vector has in (4), but they are fewer; moreover, such approach makes it possible to track the equations that determine the modelling success or failure).

The modelling procedure is stopped when (and if) a desired accuracy is reached for each equation $h_i(\mathbf{x}, \mathbf{y}) = 0$ in the set (2).

The goal of design optimisation here is to find the set-point $[\mathbf{x}^\star, \mathbf{y}^\star]$ for an existent power plant where the performance index is minimised (cf. App. A). A set-point in the considered model is a vector of model variables: flows of fuel (coal), working factor (water) and electric current, pressures, temperatures, entalpies and helper coefficients. There are 539 such variables in the considered model, which describe state of 67 model elements.

The performance index is the difference of earnings from selling electric energy and costs of the fuel. It must be noted that electric energy is just a by-product here: the major objective is to supply the factory with the steam of desired parameters. However, producing steam is raison d'être of the plant, so there is no point in treating steam parameters as the decision variables.

The need to adjust the plant working point so that it operates at the smallest cost possible appears several times a day, mainly because of the changing steam demands following the production schedule, usually known in advance. The optimal set-point location may also be affected by changes of price of electrical energy.

In the model, there are 21 decision variables, selected by a skilled engineer so that solving (2) is done efficiently. They are subject to box constraints (3a). The main purpose of those constraints is to curb the range of decision variable values in attempt to approxi-

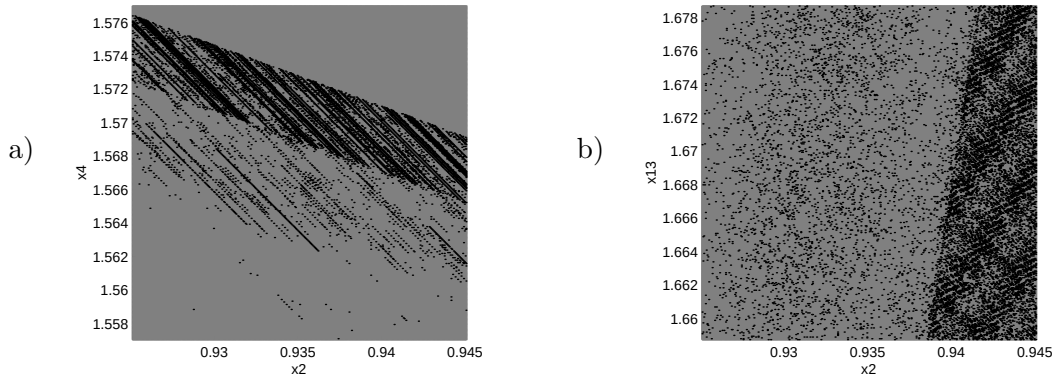

Figure 1: Two-dimensional exemplary cuts through $D_{\mathbf{x}}$ with sets of decision variables values for which the modelling (i.e. computing of $\mathbf{y}$ by simulation) fails marked with black colour.

mate roughly the set a trained human operator would consider the reasonable for model input values.[2]

While inspecting the model behaviour, important adverse features of both constraints and objective function have been found that hinder efficient problem solution seeking. They were reported earlier in (Kamola and Malinowski, 2000). Regarding constraints, those implicit ones defined by existence of a solution of (2) have very irregular shapes. Some of them are shown as black areas in Fig. 1 where exemplary cuts through search domain $D_{\mathbf{x}}$ are drawn. They are in form of disconnected sets or isolated points, and are extremely difficult to handle by any optimisation routine.

Implicit constraints defined by (3b) are not so numerous and nasty: they appear to lead to convex but still disconnected feasible sets. They are shown in Fig. 2 in dark gray colour. It is interesting to point out that the areas of feasible points (light gray) can adjoin areas where the simulation fails altogether (black — non-existence of a solution to (2)). This is particularly inconvenient because the optimisation procedure may step from "safe" area directly into a "minefield".[3] None of these unpleasant features of the problem matter

[2]Moreover, upper and lower bounds on $\mathbf{x}$ bracket the nominal values of the corresponding internal model variables, and so determine the region of numerical model validity.

[3]Non-existence of a solution to (2) is manifested in practice by the modelling module by reporting a numerical exception, or entering an infinite loop. It is practically impossible to judge whether such action is the effect of errors in model implementation, errors of modelling or represents forbidden states of the object and denotes real emergency situations — all cases are probable.

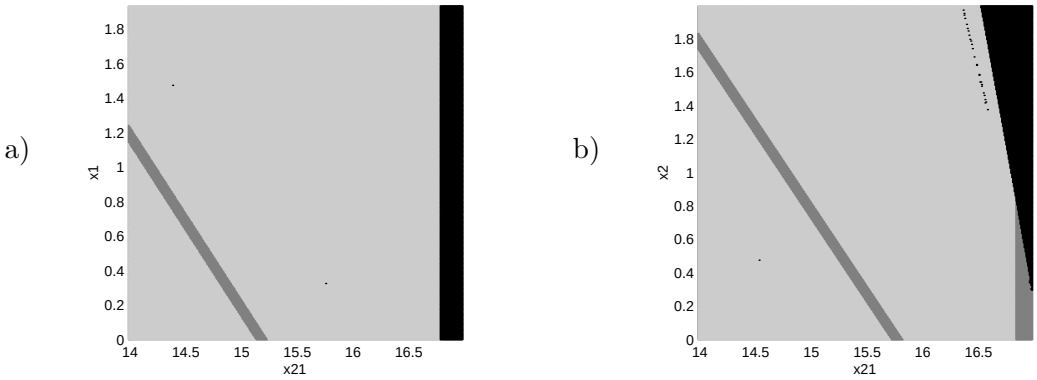

Figure 2: Other cuts through $D_{\mathbf{x}}$ where feasible regions are drawn in light gray, regions with implicit constraints (3b) violated drawn in dark gray, and regions with implicit constraints (2) violated marked in black.

too much when the design optimisation is to be performed by a user well learned into the specifics of the modelled plant. In fact, the modelling module was made with intention to be operated manually by such a user. While a specialist avoids dangerous or nonsense combinations of design variables values, an automatic optimisation routine wanders in every corner of $D_{\mathbf{x}}$, causing modelling failures. However, replacing a human operator with an optimisation algorithm is desirable in the aftermath as it never gets tired, moreover, by operating in a non-routine way it can discover completely new solutions.

The adverse features of the objective function mentioned above are the harshness and sudden steps of the response surface, as shown on directional graphs of $f(\cdot)$ in Fig. 3 and Fig. 4, respectively. There is nothing unusual about them in design optimisation; however, together with irregularity of constraints, they effectively prevent the use of gradient-based optimisation. The harshness or simulation noise is caused by finite accuracy termination criteria in loops calculating $\mathbf{y}$ for given $\mathbf{x}$. The stepwise character is the effect of switching in modelling formulae which, for example, is the result of a transition of working factor between fluid and gas states.

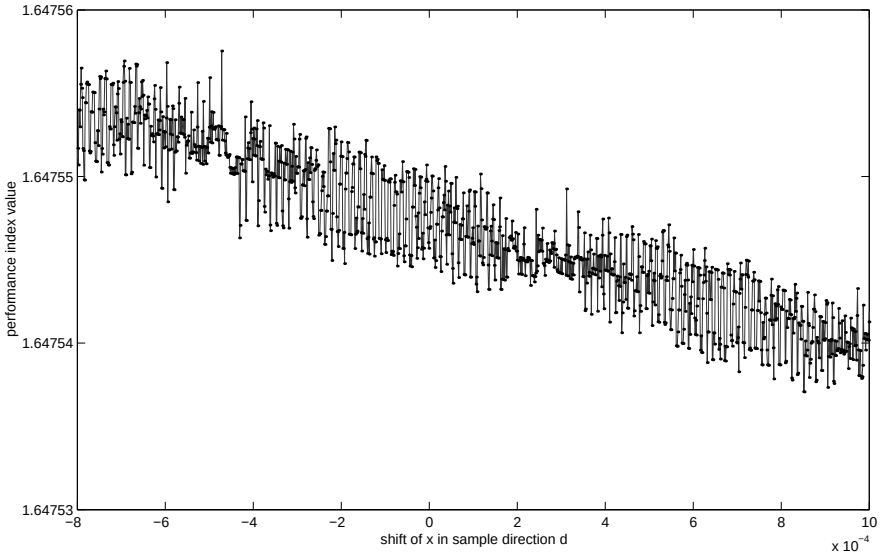

Figure 3: Graph of performance index computed close to the problem optimum along a given direction **d**.

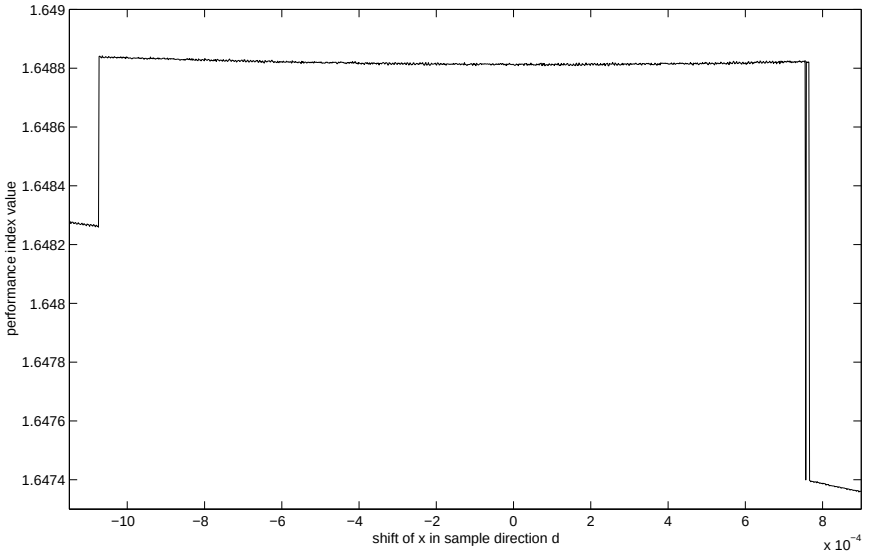

Figure 4: Stepwise character of $f(\cdot)$ shown on directional graph drawn near the problem solution.

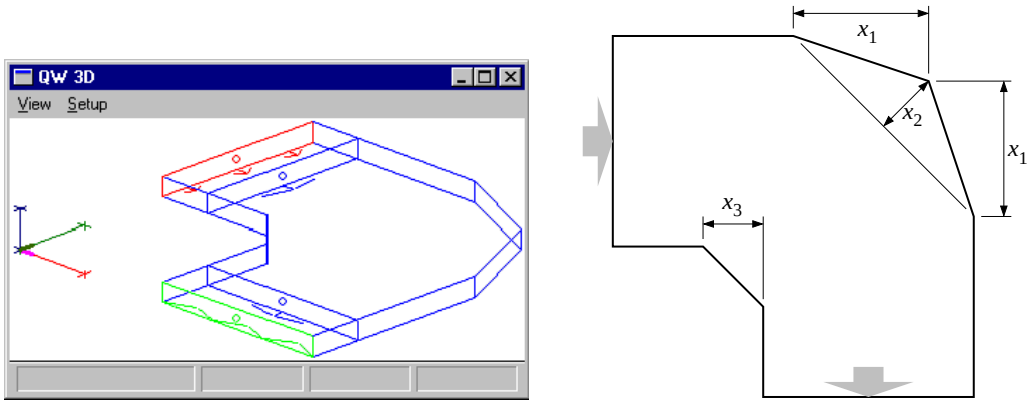

Figure 5: Three-dimensional (left) and top (right) view of the waveguide bend subject to design optimisation.

3.2 Shape optimisation of a microwave guide

The process of microwave circuit design optimisation is concerned with adjusting waveguide dimensions with the purpose to obtain an element with desired electromagnetic properties. Analysis of the designed microwave circuit behaviour is performed nowadays rather by electromagnetic field simulators than by using field theory open formulas. Simulators make it possible to calculate the properties of microwave elements of fancy shapes.

The modelling is done using finite difference time domain method (Taflove, 1995), where both space and time domains are discretised; next, discrete Maxwell equations are employed to compute electromagnetic field distribution in every single cell of the partitioned object and in every single time instant. Therefore, simulation speed (not so much as accuracy) depends dramatically on discretisation. Choosing appropriate temporal and spatial discretisation patterns is essential to make the simulation tool practically usable.

In the particular optimisation problem considered here (Kamola and Miazga, 2001), three dimensions x_1, x_2, x_3 of a waveguide bend are the decision variables, as shown in Fig. 5. Waveguide is a kind of transmission line, frequently used in gigahertz frequency devices, and transmitting quite big energy — like in the case of radars. Waveguide walls are made of metal and the air inside is the media the wave propagates through.

The objective of the optimisation process is to find the design in which the wave reflection coefficient of the waveguide stays small within a specific range of frequencies. If the output vector containing reflection coefficients for frequencies in the range of interest is the simulation output $\mathbf{y}$, then the performance index is

$$f(\mathbf{y}) = \|\mathbf{r}(\mathbf{y})\|_{L_p} \quad , \tag{5}$$

where $\mathbf{r}(\mathbf{y})$ is a linear penalty function, activated if the reflection coefficient raises above some threshold. The penalty function $\mathbf{r}(\mathbf{y})$ computes a vector of penalties for all frequencies in the considered range, $f(\mathbf{y})$ is a L_p norm of $\mathbf{r}(\mathbf{y})$.[4] To complete the optimisation problem definition, $\mathbf{x}$ is subject to simple cubic constraints.

Unlike in the power plant problem, there is no way for the electromagnetic field simulator to fail, an so the implicit constraints (2) are always satisfied. Also, the other sort of implicit constraints (3b) vanishes, incorporated into penalty function $\mathbf{r}(\cdot)$. Therefore, the problem poses no difficulties related to constraints. However, the shape of performance index itself worsens as p increases. In the most desirable case of $p = \infty$ (i.e. when there is no indulgence even for the slightest violation of (3b)), the response surface is corrupted with noise and steps mixed, as shown in Fig. 6. As in the case of power plant model, the noise is the effect of numeric inaccuracies and computations being carried out mostly in loops. The stepwise character of $f(\cdot)$ is the effect of space discretisation: increasing some detail size causes new cells to be created automatically — at this little "catastrophe" the instant performance change is observed. The finer is the discretisation, the bigger number of steps is observed, their size decreasing.

[4] $\|\mathbf{v}\|_{L_p}$ for an even $p \geq 2$ is defined for some vector $\mathbf{v}$ as follows

$$\|\mathbf{v}\|_{L_p} = \begin{cases} \max_{i \in \{1,\ldots,\dim \mathbf{v}\}} |v_i| & \text{when} \quad p = \infty \\ \sqrt[p]{\sum_{i=1}^{\dim \mathbf{v}} v_i^p} & \text{else.} \end{cases} ,$$

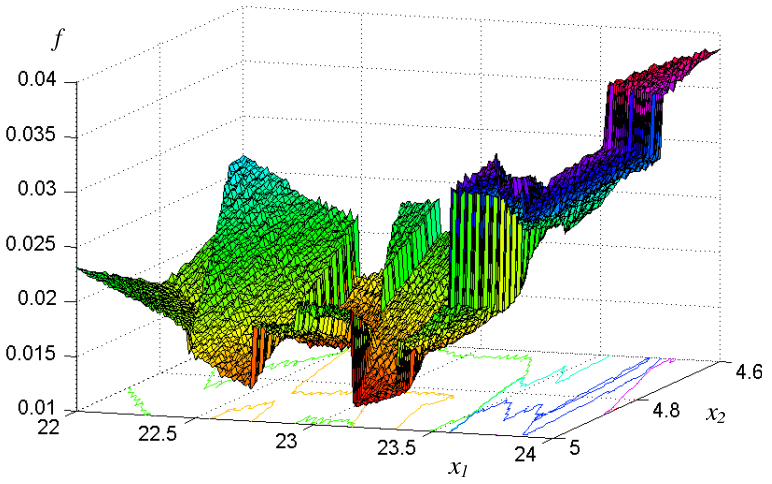

Figure 6: Fragment of performance function surface for waveguide bend design optimisation problem.

3.3 Proposed methodology for solving design optimisation problems

It has been shown that unpleasant properties of design optimisation problems may be rooted directly in the nature of the system, they may be the modelling artefacts, or they may result from sheer model implementation errors. What to do about them if they are really a serious obstacle for any optimisation routine? Many authors (Papalambros, 1988; Hammel, 1997) demand that the models must first be verified and simplified before the optimisation process can start. However, it seems from the author's experience that changing the way the modelling is done is in many cases impossible. This is either because the modelling routine is contained in an off-the-shelf analysis tool, or its intricacies are so deep that tracing errors and inaccuracies (e.g. by reverse engineering or in any other way) would take ages. Therefore, an optimisation engineer has his/her hands tied — at least as far as operation of the simulation module is concerned.

Admittedly, the designer cannot change the way the modelling is done, but he can play with the optimisation strategy or with the system modelled. These are the fundamental assumptions contained in the work (Kamola, 2004). Let us examine these two possibilities in the context of current trends in design optimisation.

It follows from Table 1 that the user is usually not allowed to provide his own optimisation procedure, neither can the user prescribe which optimisation routines built into the tool will be used in the course of optimisation, and in which order.[5] As regards models used in the course of design optimisation, the standard approach is to apply a coarse-grain one for preliminary, and a fine-grain one in the final stage of optimisation. This is a standard approach, and definitely a working one. For instance: use non-gradient optimisation routine with the rough, and local gradient optimisation routine with the fine model. However, how one can know precisely which routine is most adequate in each stage of optimisation process? The existing design optimisation packages require the user to use his own brains and problem knowledge to specify that. However, in many practical cases the user buys an off-the-shelf analysis tool and, striving to couple it with an optimisation solver, is perplexed because the user's actual knowledge of the nature of the problem generated by the modelling tool can be almost none. Therefore, the user does not know which optimisation routines to use, in which order, and what should be the termination criteria there — so that desirable solution quality will be attained in reasonable time. The example of waveguide design is in place to be mentioned here: as the end user (say, the chief designer in microwave devices manufacture) acquires a modelling software, he/she expects his/her team to embed it into an optimisation routine quickly and effectively. Possibly none of those people knows much enough about the modelling tool (neither about the optimisation theory) in order to be able to utilise this knowledge and to choose the optimisation routine matching the best.

The methodology proposed in this paper tries to help the user with the construction of optimal hybrid optimisation routine. It proceeds as follows:

1. Use as much a priori knowledge of the problem as available in order to choose a set of

[5] Epogy is one of the very few exceptions to this rule; its operation and results will be used as a reference point in Section 4.

candidate optimisation routines capable to solve the problem jointly. They should be few and simple enough to allow for some ad hoc modifications. If nothing of the problem is known, choose several direct search global methods to be run in the preliminary stage of optimisation, and some gradient-based local search algorithm to be run close to the optimum.

2. Define a system that is simpler than the original one (e.g. by taking away a part of the original system) but preserving the original system features. Use the original modelling tool and methodology to model it.

3. Use the selected optimisation routines to find the solution of the simplified problem, and to determine the combination of those that constitute the most efficient and effective hybrid optimisation tool. (This will take much less time than for the original problem as the dimensionality of the simplified model is smaller).

4. If any unpleasant features of the simpler problem appear, make necessary improvements to the optimisation routines so that they are capable of solving this problem. If this does not help, look for some other optimisation algorithms. (Simplicity of the algorithms in the candidate set will allow rapid modifications).

5. Work out efficient criteria for switching between routines qualified to be combined into the hybrid algorithm.

6. Apply the optimisation algorithm made for and trained on the simplified problem, to the original problem.

This methodology does not introduce any revolutionary concept, but lets the engineer inspect problem features while making the optimisation procedure, which — applied to the original problem — should work properly without too much extra effort spent. Section 4 present results of this methodology application to problems described in Section 3.1 and

in Section 3.2. Both the standard and the proposed above methodologies are presented in the form of flowcharts in App. B.

4 Results

4.1 Power plant example

Initial selection of candidate optimisation methods. This selection was affected by what has been known so far about the specifics of the modelling tool used, and by the way the modelling was done in competitive design optimisation packages for heat and electric systems. Those packages, in general, adopt much simpler models and are capable of modelling much simpler structures (e.g. turbosets not interlinked by common steam collectors, unlike ours) — and consequently use simple, sometimes linear, optimisation methods. On the other hand, the modelling tool used employs highly nonlinear formulae. Eventually, two global procedures have been qualified for preliminary optimisation and one gradient local procedure — for final optimisation stage.

Those global methods are: controlled random search (CRS2) and evolutionary algorithm (EA). Both are direct search routines that operate by transforming a pool of trial solutions so that they finally focus on the global solution with great probability. Generalised reduced gradient (GRG) method has been chosen for final optimisation as it offers efficient handling of implicit constraints (2) without violating them at the intermediate phases of its operation. The major competitor, sequential quadratic programming, was discarded mostly because it relies on quadratic approximations to $f(\cdot)$ (that may be highly inaccurate) and because it violates constraints (2) in the course of operation. By this choice, it was almost sure that the final hybrid algorithm will discover globally optimal set point, while being efficient in tracing it precisely. The question was only which global routine to choose, and when to stop it and activate the GRG.

Definition and modelling of a simplified system. The simplified system (cf. App. A) was created by excluding three one-stage turbines and the accompanying devices, like regeneration system or collectors, from the original system. All pumps have been removed too. However, most nonlinear elements, i.e. turbines (and, potentially, boilers) were preserved as well as the performance index formulation. System simplification reduces the search space dimension from 21 to 9, and the average simulation time from 0.5 to 0.2 sec, which makes it possible to experiment quite freely even with time consuming non-gradient optimisation routines.

Selection of the best optimisation methods. Numerous tests have been performed with purpose to find best optimisation settings for CRS2 and EA algorithms. They are reported in detail in (Kamola, 2004, pp. 87–94). Main conclusions are:

- Discarding trial points for which the simulation fails does not make difference for efficiency from the case of keeping them in the algorithm pool, marked with very high objective value. This applies for both CRS2 and EA.

- Enhancing algorithm exploratory nature (by increasing point pool size in CRS2 and by increasing variability of mutations in EA) comes at the cost of much computational effort. Both algorithms are capable of finding an optimum if the computational budget is high.

- Neither of the two algorithms is suited to carry out the optimisation by itself alone, as the optimal solution is situated on (3b) constraint (cf. Fig. 7), of the type not being supported well neither by CRS2 nor EA.

- When run with cost-efficient settings (small pool point, little explorability), both CRS2 and EA find feasible solutions. It seems that the solution quality depends mainly on the number of objective evaluations, not on the algorithm type. See Table 2.

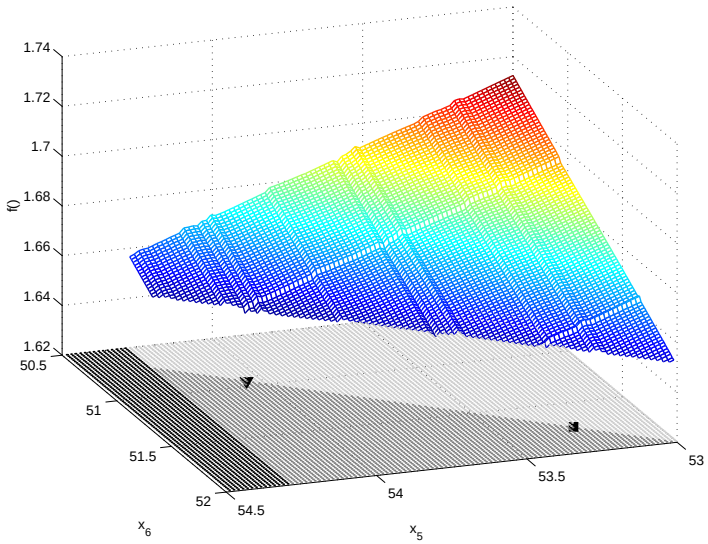

Figure 7: Reference locations of the simplified problem solutions in the search subspace of x_5 and x_6 and the performance index surface. (CRS2 solution location is marked with a triangle; Epogy solution is marked with a square.)

accuracy ϵ_A	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
CRS2	**1.764**	**1.720**	**1.733**	**1.850**	**1.855**	**1.876**
solutions	*100000*	*72071*	*36893*	*2969*	*927*	*442*
EA	**1.690**	**1.694**	**1.693**	**1.694**	**1.765**	**1.997**
solutions	*94413*	*99075*	*96613*	*99050*	*55400*	*5638*

Table 2: Average performance index (boldface) values and number of $f(\cdot)$ evaluations (italics) for simplified problem solving by CRS2 and EA. Termination criterion was the relative accuracy ϵ_A (subsequent solutions differ by ratio not more than ϵ_A).

Based on such observations, CRS2 was selected as the preliminary routine. It was decided so mostly on the account that it has a very short "cycle" of producing new solutions, so it is easy to verify quickly whether to switch to next routine in the hybrid optimisation algorithm. However, there is no strong argument behind such a decision; probably an extra CRS2 feature acting in favour of it is that the results are more predictable and in line with common sense (tougher termination criterion: more function evaluations — better solution obtained).

Modification and reselection of methods. The problem of simulation failures encountered — and worked around — while applying the two direct search methods was considered to be the main obstacle in applying GRG. Fortunately, a solution neighbourhood, i.e. the region in which gradient methods should be applied, was free of such peculiarities. However, numerical experiments aiming to make objective gradient estimates using finite forward difference scheme with adaptive step size selection showed great gradient variability (cf. Fig. 3). Consequently, cursory optimisation tests made with a simple sort of steepest descent scheme have failed altogether, and the idea of GRG application had to be replaced by something else.

COMPLEX algorithm (Box, 1965) was then considered for the final optimisation stage. It is a direct search algorithm (its name stands for "constrained simplex"), and it has direct support for constraints in the form of (3b). Its only drawback is that it is designed for convex search domains, let alone being able to support simulation failures. Changes were necessary in order to make it applicable to our case, and not only in the close neighbourhood of the optimum.

In case of using COMPLEX, similarly to CRS2 and EA, the optimal solution is sought by transformations of the pool of trial points, by means of worst point reflection w.r.t. some reflection centre. In case of implicit constraint violation, the standard remedy in COMPLEX is to move the reflection centre toward the pool centre of gravity. The modi-

w	100	50	20
original	**1.862**	**2.037**	**2.111**
version	*608*	*286*	*180*
improved	**1.759**	**1.786**	**1.897**
version	*483*	*319*	*190*

Table 3: Average performance index value (boldface) and number of $f(\cdot)$ evaluations (italics) for simplified problem solving by COMPLEX routine without and with the author's improvements. (Parameter w is explained in *Switching Criteria* paragraph.)

fication proposed by the author is based on the observation that such gravity centre may be infeasible — but the best solution found so far *is* feasible, and thus it has been made the reflection centre of last resort, approached gradually when everything else fails.

Such changes made the procedure robust enough: undesirable phenomenon of algorithm getting stuck before reaching a solution has been eliminated as shown in Table 3, where average performance indices and the accompanying number of algorithm steps are shown. One may draw the conclusion that such a modification is a prerequisite for any qualitative improvement of a solution. It can also improve efficiency, especially if only mild accuracy is required.

Switching criteria. Finally, the hybrid procedure consists of CRS2 being run with pool size of $8(\dim \mathbf{x} + 1)$, that is about the minimum pool size suggested in the literature. Then COMPLEX takes over, running with pool size of only $2(\dim \mathbf{x} + 1)$ and terminating when the objective value has not changed within the last $w = 50$ algorithm steps by more than $\epsilon_{\mathrm{I}} = 0.001\%$. Such parameter values result from tests reported in detail in (Kamola, 2004, pp. 97–101).

In view of CRS2 inefficiency and big variability of both number of iterations and of solution quality, it was proposed that the switchover criterion be simple and not based on current CRS2 efficiency. Consequently, CRS2 operation was terminated after a given number of objective function evaluations, and COMPLEX was started from the best so-

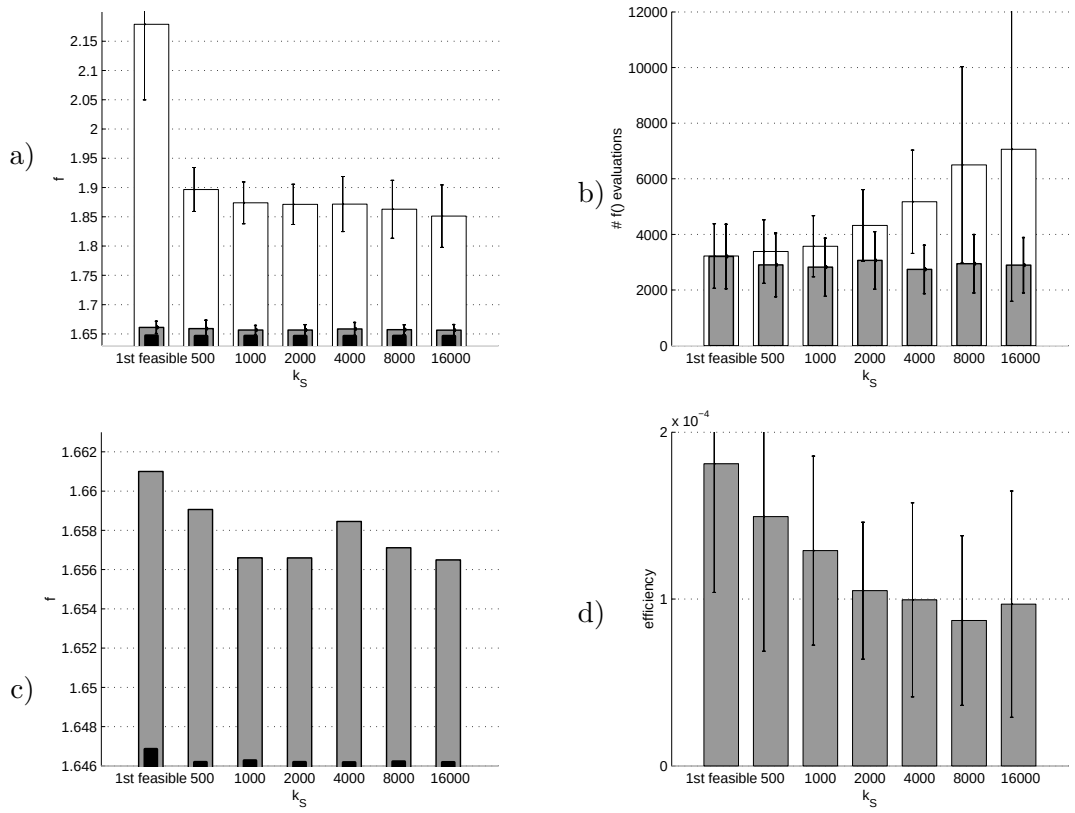

Figure 8: Selected statistics for operation of CRS2/COMPLEX hybrid optimisation algorithm. Bars indicate average values; lines attached to bars indicate standard deviations from the average values. Results are drawn for varying switching criterion k_S — the number of $f(\cdot)$ evaluations after which a switchover from CRS2 to COMPLEX is performed.
Graph a: performance index value at the solution found by CRS2 (white bars) and COMPLEX (gray bars); also the best result in a series of algorithm runs (black bars). **Graph c:** closeup of graph a. **Graph b:** total number of $f(\cdot)$ evaluations for COMPLEX (gray bars) and for the whole hybrid algorithm (white bars). **Graph d:** optimisation efficiency (improvement in solution quality divided by the number of $f(\cdot)$ evaluations) for the whole algorithm.

lution[6] found so far. Solution qualities, number of objective evaluations and the resulting efficiency are shown on graphs in Fig. 8. Also, the switchover after the first feasible solution is found is included as an extra criterion. It is evident (graph *a*) that increasing CRS2 budget substantially has very little impact on CRS2 solution quality, which is in turn even less correlated with the final solution quality (graph *c*). Consequently, with budget growing (graph *b*), the total efficiency is decreasing (graph *d*). The conclusion can be drawn that CRS2 should be run until it finds any feasible solution; then COMPLEX should be used to do the rest of the job. This may mean that the considered problem is

[6]Starting COMPLEX with only the best point found by CRS2 or with a whole content of CRS2 point pool (as an extra information) does not have any effect on COMPLEX solutions quality.

not truly multimodal, and its major difficulty lies in an efficient treatment of simulation failures, surface harshness and implicit constraints. It must be mentioned that graphs in Fig. 8 present mean values from 100 optimisation runs; by observing standard deviations one may conclude that COMPLEX results also vary much from one run to another, and running several instances of the algorithm — if parallel computation environment is available — would be the best strategy. The results obtained for the simplified model are comparable to those produced by a commercial design optimisation tool.[7]

Solving the original problem. The tandem CRS2/COMPLEX has been applied then for solving the original full-size problem. The only changes in the algorithm parameters concerned the change of point pool size, and setting history window w to 120. It must be stressed that the algorithm finds the problem solution reliably and efficiently with no need of any further changes. The solution is reasonable in the opinion of specialists. Such judgement is based, among other factors, on flow distributions at the solution: less efficient boilers receive as little water as possible in order to maximally reduce the use of this part of installation, and the flows through reduction valves are reduced to zero, which means that all industrial steam is first used for electricity generation before reaching the factory outlets.

The original problem was not solved by the commercial design optimisation tool because the evaluation release available to the author did not support problems of such dimensionality. Therefore, no final verification of the author's methodology effectiveness and efficiency was possible at that stage.

[7]The objective function has the value of 1.646 at the best solution found by CRS2. In turn, the Epogy package has found a solution with performance value of 1.650. Such value has been found after some 25,100 evaluations of $f(\cdot)$, and no further improvement was detected in the following 20,000 evaluations. Like for CRS2, Epogy solution is also located on the brink of implicit constraints area (cf. Fig. 7), nevertheless Epogy professional optimisation solver was apparently not able to 'glide' along it in pursue of a better point. Such outcome of the optimisation experiment strengthens the general observation that solving simulation-optimisation problems is a tough task, in spite of a number of worked out pre- and post-optimisation techniques.

4.2 Waveguide example

Initial selection of candidate optimisation methods. The reason for which the waveguide design is formulated as a design optimisation problem is that it has become too troublesome to operate the simulation tool manually — exactly as in the case of power plant problem. Replacement of the operator's experience and intuition with unemotional operation of an algorithm finally turns out to be more economic. However, it is still the engineer that decides which optimisation routine to use, and which solution is to be accepted.

In case of waveguide there were earlier trial applications of Powell optimisation routine (Press *et al.*, 1992, pp. 412–420). However, Powell was found incapable of performing the whole optimisation process as it appeared to be very sensitive to starting point location, and to stepwise response surface character. Therefore, it was designated as a candidate for an optimisation routine to be used at final stage of optimisation. CRS2 was chosen again for performing the initial optimisation. Such choice was biased by good CRS2 performance in the case of power plant problem, and by inadequate performance of a competitive EA algorithm, an opinion shared by the specialists so far operating the stimulator manually.

Definition and modelling of a simplified system. To accelerate the process of optimisation, the dimensionality of search space was decreased to two by fixing the value of decision variable x_3. It defines the width of a chamfer used for compensation of so-called fringing field effects which occur at sharp metal edges (see Fig. 5).

Selection of the best optimisation methods. In the case of waveguide design there was no possibility to test freely and compare several optimisation methods, since the simulator was a commercial product controlled by a company, and the design optimisation plug-in was to become a part of it. Therefore, the possibilities to use the simulator at wish, and to change the working of optimisation routine, were limited.

Separate tests of CRS2 and Powell routines have shown that CRS2 needed more eva-

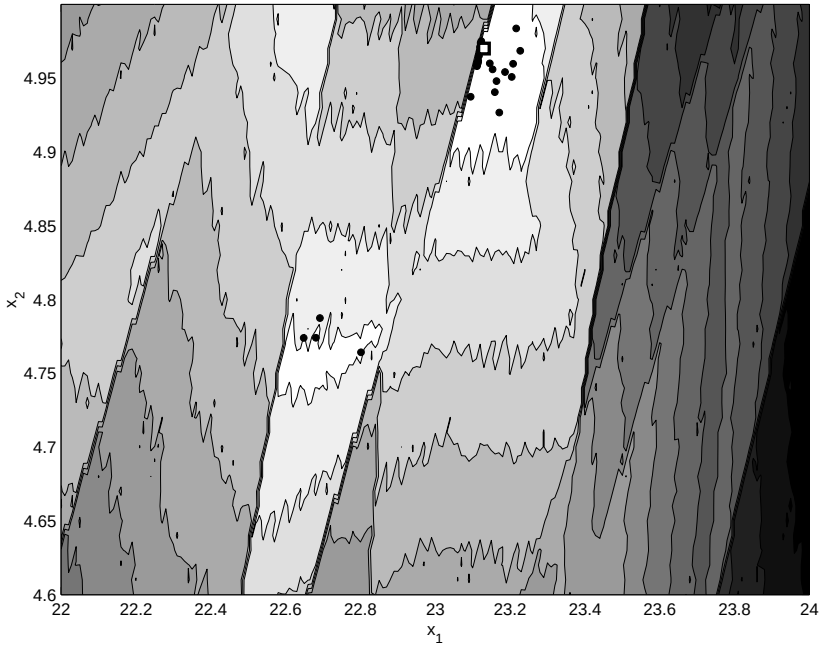

Figure 9: Location of CRS2 (black dots) and Powell (white square) solutions for the simplified waveguide design problem on the contour plot of performance index. White areas denote regions of good performance (low value of the performance index).

luations in a single run than Powell, but it found satisfactory solutions much more reliably. On average, CRS2 run alone needs 30% less computational budget to work equally efficiently to Powell. Such result seemed satisfactory, and CRS2 was appointed to do the whole work alone. Therefore, *Reselection of methods* and *Switching criteria* parts of the proposed procedure were no longer applicable. Fig. 9 presents location of several CRS2 solutions, and of the best Powell solution. CRS2 solutions are sometimes located in proximity of the local, but prevalently in proximity of the global minimum. Powell method started from a location chosen at random is capable of finding a good solution — but this is a rare case.

Modification of methods. Those were directed to improve CRS2 efficiency in two ways: by changing the rules of trial point reflection, and by making the algorithm run in parallel. If a reflected trial point was found to violate constraints (3a) it was no longer cast on the search domain but "bounced" at the domain boundary back into the feasible

region. This prevented CRS2 pool from "flattening" at the boundary. The parallel version of CRS2 (Kamola and Miazga, 2001) is not based on suggestions found in the literature (Price, 1987); instead, it maintains a common pool of points, and parallel execution threads use first, second, etc. worst points in the pool, producing simultaneously as many trial points as is the number of active threads. Such solution promotes rather explorability than efficiency — but so modified CRS2 has been found to work quickly enough and reliably.

Solving the original problem. To verify quite peculiar observations and assumptions about CRS2 capabilities a number of optimisation trials has been performed for the original 3-D problem. Powell and CRS2 methods were started from points randomly selected from the domain. Out of 100 optimisation runs, the number of successful ones, i.e. with performance index value reduced to below 0.012, was 8 for Powell and 49 for CRS2 routine. Respectively, the average number of $f(\cdot)$ evaluations after which algorithms stopped was 167 and 1000.[8] Let us calculate the number N of objective function evaluations needed to find a satisfactory solution with probability p:

$$N = n \log_{1-r}(1-p) \quad , \tag{6}$$

where n is the average number of $f(\cdot)$ evaluations made by an algorithm and r is the ratio of good quality solutions obtained in a series of optimisation runs. Smaller value of N means more efficient method. Using (6) as efficiency measure, we find Powell ($N = 6000$) still less efficient than CRS2 ($N = 4449$) in the 3-D case.

However, increase of the required computational budget (due to the increase of the problem dimensionality) made it possible to reconsider application of a hybrid CRS2/Powell routine. Fifty optimisation runs of such a routine have been made, where Powell was

[8] Alternatively, one could use the actual number of $f(\cdot)$ evaluations after which the solution was found — but this number is known only when the stop criterion is satisfied. The stop criterion was set arbitrarily by a trained operator, and reflects both his knowledge of Powell behaviour and disbelief for CRS2 applicability.

activated after $k_S = 130$ or after $k_S = 260$ function evaluations made by CRS2. The

	algorithm	Powell only	hybrid ($k_S = 130$)	hybrid ($k_S = 260$)	CRS2 only
1.	average number of $f(\cdot)$ evaluations	167	223	338	1000
2.	percentage of good solutions	8	38	58	49
3.	N, efficiency	6000	1397	1167	4449

Table 4: Statistics for 3-D waveguide design problem solved by various algorithms. 1. Average number of $f(\cdot)$ evaluations calculated for those algorithm runs in which the solution was satisfactory. 2. Percentage of such satisfactory algorithm runs. 3. Efficiency estimated using (6), for the actual number of iterations made.

results are given in Table 4, along with the reference performance of CRS2 and Powell, run alone. It turns out that by applying a hybrid routine one can significantly reduce the effort needed to obtain a good solution with acceptable confidence.

4.3 Concluding remarks

The main conclusions of this paper are the following:

1. Complex design optimisation tasks are solved best by hybrid optimisation routines, consisting of a global search algorithm and a local search algorithm, activated by some robust switchover criterion (e.g. after executing a predefined number of steps by the global algorithm).

2. Selection of the component algorithms can be done using a simplified model of a system, i.e. simpler than the original model (especially in terms of dimensionality) but preserving features of the original system that are important for the optimisation process (character of constraints and of response surface etc.).

3. Preferring simple optimisation algorithms pays unless non expected and adverse problem features appear, as it is much easier to construct ad-hoc workarounds.

4. The recipe for making an efficient and robust optimisation routine can roughly be

presented in form of some design procedure; making the simplified model is both engineering and art, not subject to algorithmisation.

The first conclusion was confirmed by the final results for both example problems. The applicability and benefits of an approach outlined in conclusion 2 is best visible in case of power plant model. Conclusion 3 is justified by both examples: avoiding gaps in feasible domain or parallelising an optimisation algorithm was particularly simple for COMPLEX and CRS2 direct search routines. The last conclusion is also supported by both examples: in case of power plant, eliminating from the original model a part of the installation just reduced the number of complicated formulas, but it did not eliminate any formulas of a particular *type*. In case of waveguide the true effect of the problem simplification was not tested in all respects for objective reasons; however, from the preliminary tests (not presented here) concerning similar models of higher dimensionality it turns out that CRS/Powell tandem efficiency decreased rapidly. This may mean that conclusion 2 may be applicable only for selected types of design optimisation problems. Which ones? — this is to be investigated.

Finally, the basic fact should be emphasised that the proposed methodology is really useful for solving design optimisation problems that are considered to be very difficult. It does so not by designing optimisation routines resulting from in-depth problem analysis; nor does it so by the application of brute force — it leads another way in the middle between those extreme approaches, somewhere besides current standard approach to design optimisation problems. Additionally, it gives an engineer the full control and freedom when composing and operating the hybrid optimisation routine, which in general should not perform worse than professional integrated design optimisation tools.

Acknowledgments

The experiments with power plant set-point optimisation example were carried out using Symul modelling package (Bujalski, 2001), developed in the Institute of Heat Engineering, Warsaw University of Technology. The author wishes to thank Wojciech Bujalski, PhD, for any help and expertise.

The experiments with microwave bend optimisation example were carried out using QuickWave (QWE, 2003a; QWE, 2003b) modelling package, developed and made available to the author by the courtesy of QWED, Ltd. The author wishes to thank Przemysław Miazga, PhD, (Institute of Radioelectronics, Warsaw University of Technology) for any help and expertise.

A Description of the power plant model

In a coal power plant the energy from coal combustion in a boiler is received by water, which is the working medium. The water, in the form of steam, goes through other devices of the plant, giving away its energy and changing its parameters. The most complex receivers of steam are turbines of turbosets. In a turbine the steam gradually decompresses and cools down on a series of propellers in the three turbine parts (high-, medium- and low-pressure), spinning them. Usually, it is possible to let out some steam of various parameters at several extractions along its passage through the turbine. The output from a turbine are mechanical power, converted subsequently by the generator into electricity, and heat energy carried by steam flows of different parameters. The steam, to be warmed again, has to be regenerated, i.e. condensed in condensers, cooled down in heat exchangers, deareated and pumped, under high pressure, again into the boiler. Big systems may consist of many such power blocks (boiler–turbine–regenerator), interconnected through collectors to ease working medium distribution in order to react

to e.g. changing power demand or to failures.

The diagram of the considered plant is given in Fig. 10. Efficiencies of selected devices are given in italics. Flows that are decision variables are marked with ⧗, with the decision variable index given beside. Industrial steam outlets are marked with ■. Flows or power levels contributing to positive $f(\cdot)$ component in (7) are marked with •, and those contributing negative component — with ∘. The factory needs three types of steam of exact pressure, temperature and flow, and one of partially adjustable flow. Steam of desired parameters comes from taking and mixing steam from various points in the installation. The factory management is interested in minimising running costs of the power plant by setting its steady-state working point appropriately. The running costs are defined as simple balance of cost of coal consumption, cost of pumps operation and the gains from selling the electric power;

$$f(\mathbf{x},\mathbf{y}) = c_{\mathrm{C}} \sum_{i=1}^{\dim \mathbf{y}_{\mathrm{C}}} y_{\mathrm{C},i} + c_{\mathrm{E}} \left(\sum_{i=1}^{\dim \mathbf{y}_{\mathrm{P}}} y_{\mathrm{P},i} - \sum_{i=1}^{\dim \mathbf{y}_{\mathrm{E}}} y_{\mathrm{E},i} \right) \quad , \tag{7}$$

where $\mathbf{y}_{\mathrm{C}}^T = [y_{\mathrm{C},1}\ y_{\mathrm{C},2}\ \ldots]$ is the vector of coal flows (expressed in kg/sec), $\mathbf{y}_{\mathrm{P}}^T = [y_{\mathrm{P},1}\ y_{\mathrm{P},2}\ \ldots]$ is the vector of power consumption by plant pumps (expressed in kilowatts), $\mathbf{y}_{\mathrm{E}}^T = [y_{\mathrm{E},1}\ y_{\mathrm{E},2}\ \ldots]$ is the vector of power levels in generators, and c_{C} and c_{E} are coal and electricity prices: 0.3 PLN/kg and $5.555 \cdot 10^{-5}$ PLN/(kW·sec), respectively.

The diagram of prepared reduced dimensionality plant model is presented in Fig. 11.

B Flowcharts for the standard and the proposed optimisation approach

The schemes of execution for both the usual design optimisation procedure, and for the one proposed in this paper are presented in Fig. 12. Most of the reasoning and the decision procedures are presented there as mere functions. This is because one can easily identify

their input and output; however, they can hardly be considered Turing machines as their effective execution requires in many cases human reasoning.

Symbols common for both procedures:

Π — a set of vectors containing numeric values, $\Pi = \{\pi_1, \pi_2, \ldots, \pi_{\bar{\Pi}}\}$. Those vectors are the formal way of presenting any kind of information collected in the process of design optimisation. In particular, they include optimisation constraints, trial points and solution approximations location, values of the performance index, and the information of the context in which it has been obtained (e.g. index of the optimisation method used, step number etc.) ($\hat{\Pi}$ refers to the same kind of data for the model of reduced dimensionality.)

M — a set of functions, $M = \{\mu_1, \mu_2, \ldots, \mu_{\bar{M}}\}$. Each function represents the action undertaken in one step of an optimisation method i, i.e. $\pi = \mu_i(f, \mathbf{h}, \Pi)$. Each function produces a new piece of information π (particularly, a new solution estimation) out of a subset of information available so far.

i — the index of currently used optimisation method.

S — termination test function; $S(\Pi)$. The decision about termination of the design optimisation procedure may be based on the whole history of the process, represented by Π. ($\hat{S}$ refers to test function for termination of the optimisation of the problem of reduced dimensionality.)

T — a function that selects the optimisation method; $T(\Pi)$. The decision is presumably based on all optimisation results obtained so far. The index of the new optimisation method is the function output.

Symbols specific for the standard design optimisation approach:

$\tilde{\mathbf{h}}$ — currently used modelling function, i.e. function defining relation of $\mathbf{x}$ and $\mathbf{y}$, as in (2), where formally $\mathbf{x}$ and $\mathbf{y}$ are subvectors of π. This function gives only an approximation of the original model defined fully by $\mathbf{h}(\cdot)$.

D — the design of experiments and model update routine, $D(\mathbf{h}, \tilde{\mathbf{h}}, \mu_i, \Pi)$. It runs using original model $\mathbf{h}$, the current model $\tilde{\mathbf{h}}$ and optimisation method used μ_i, and the trial points of the domain examined so far, which are contained by Π. It chooses new trial points to be examined, calculates the performance index $f(\cdot)$ there, and updates the current model. The new modelling function and a vector of information about points examined are returned.

U — model update test function, $U(\Pi)$. It is a function determining whether the currently used modelling function needs fine tuning.

Symbols specific for the design optimisation approach proposed in this paper:

R — method repair test function, $R(\hat{\Pi})$. Used to determine whether the currently used optimisation method needs (and is subject to) repair, or just another method must be activated in order to carry out the task of optimal hybrid algorithm preparation phase. The decision is, in theory, dependent on the whole information $\hat{\Pi}$ gathered so far.

V — method repair procedure, $V(\mu, \hat{\Pi})$. The procedure produces an improved version of μ, potentially capable of dealing with discovered new adverse problem features, described by optimisation history stored in $\hat{\Pi}$.

W — procedure of preparation for final optimisation run, performed on the original model $\mathbf{h}$. $W(\hat{\Pi})$ uses all the information $\hat{\Pi}$ gained so far in the phase of trial optimisations executed on the model of reduced dimensionality $\hat{\mathbf{h}}$. Functions for effective termination criterion and method switchover are the effect of W, along with the appropriate optimisation method toolbox, M.

A — method change test function, $A(\Pi)$. Used to determine whether the next optimisation method from M can be activated.

References

Bazaraa M.S., Sherali H.D. and Shetty C.M. (1993): *Nonlinear Programming. Theory and Algorithms.* — John Wiley & Sons, Inc.

Box M.J. (1965): *A new method of constrained optimization and a comparison with other methods.* — The Computer Journal, Vol.8, No.1, pp.42–52.

Bujalski W. (2001): *Metoda rozdziału obciążeń w układach energotechnologicznych sterowanych przy wykorzystaniu systemów rozproszonych (DCS)*, PhD thesis, Warsaw University of Technology: In Polish.

Edgar T.F. and Himmelblau D.M. (1988): *Optimization of Chemical Process.* — McGraw-Hill.

Eldred M.S., Giunta A.A., van Bloemen Vaanders B.G., Wojtkiewicz S.F.J., Hart W.E. and Alleva M.P. (2005): *DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. Version 3.3. Developers manual*, Technical Report SAND2001-3515, Sandia Laboratories: Available at `http://endo.sandia.gov/DAKOTA/papers/Reference3.3.pdf`.

Hammel U. (1997): Chapter F.1.8: *Simulation models*, In: Handbook of Evolutionary Computation (T. Bäck, D.B. Fogel and Z. Michalewicz, Eds.), Institute of Physics Publishing, Bristol.

Hyperworks (2006): Altair Hyperworks: Web page available from `http://www.altair.com/software/hw.htm`.

Jankowski Z., Kurpisz Ł., Laskowski L., Łajkowski J., Miller A., Sikora W., Portacha J. and Zgorzelski M. (1972): *Model matematyczny pracy turbozespołu w zmiennych warunkach — na przykładzie bloku 200 MW.* — Biuletyn Informacyjny Instytutu Techniki Cieplnej, Vol.33, pp.3–36: In Polish (English abstract available).

Kamola M. and Malinowski K. (2000): *Simulator-Optimizer Approach to Planning of Plant Operation: Ill-Defined Simulator Case.* IFAC - MIM 2000, Symposium on Manufacturing, Modeling, Management and Control, Rio Patras, Greece, pp. 377–382.

Kamola M. and Miazga P. (2001): *Global and Local Optimization Algotithms in Automated Wavequide Design.* V KKA Algorytmy Ewolucyjne i Optymalizacja Globalna, Jastrzębia Góra, Poland, pp. 97–105.

Kamola M. (2004): *Algorithms for Optimisation Problems with Implicit and Feasibility Constraints*, PhD thesis, Warsaw University of Technology: Available from `http://www.ia.pw.edu.pl/~mkamola/Kamola04.pdf`.

Papalambros P.Y. (1988): *Principles of optimal design.* — Cambridge University Press.

Plambeck E.L., Fu B.R., Robinson S.M. and Suri R. (1996): *Sample-path optimization of convex stochastic performance methods.* — Mathematical Programming, Vol.75, pp.137–176.

Poloni C., Pediroda V., Clarich A. and Steven G. (2005): *The use of optimisation algorithms in PSO*, Project Deliverable 4, Fenet Thematic Network, Competitive and Sustainable Growth Programme: Available from `http://www.fe-net.org/technology/pso/`.

Portacha J. (1969): *Optymalizacja struktury układu cieplnego siłowni parowych*, PhD thesis, Politechnika Warszawska, Warszawa: In Polish.

Press W.H., Teukolsky S.A., Vetterling W.T. and Flannery B.P. (1992): *Numerical Recipes in C.* — Cambridge University Press.

Price W.L. (1987): *Global Optimization Algorithms for a CAD Workstation.* — Journal of Optimization Theory and Applications, Vol.55, No.1, pp.133–146.

QWE (2003a): *QuickWave-3D User's Manual.*

QWE (2003b): *QW-Optimizer User's Manual.*

Sandia (2006): Features of computationally complex engineering problems: Web page at `http://endo.sandia.gov/DAKOTA/research/complexity.html`.

Scott A.T. (2001): *An evaluation of three commercially available integrated design framework packages for use in the Space Systems Design Lab*, Report, Georgia Tech: Available at `http://www.phoenix-int.com/library/papers.php`.

Synaps (2003): *Epogy 2003. User's Guide*, Synaps, Inc.

Taflove A. (1995): *Computational Electrodynamics: The Finite-Difference Time-Domain Method.* — Boston: Artech House.

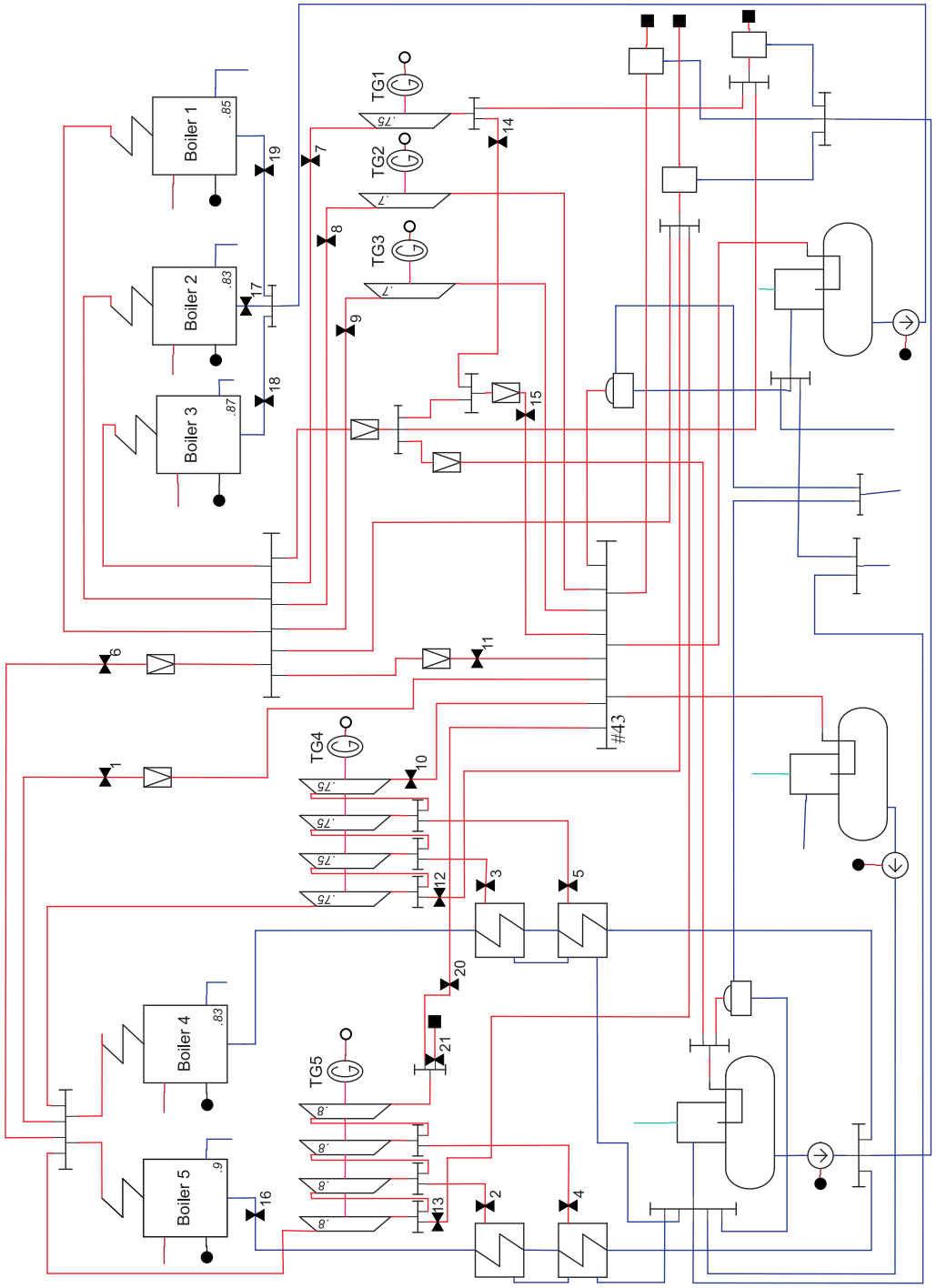

Figure 10: The diagram of industrial power plant at 'Janikosoda' chemical works in Janikowo.

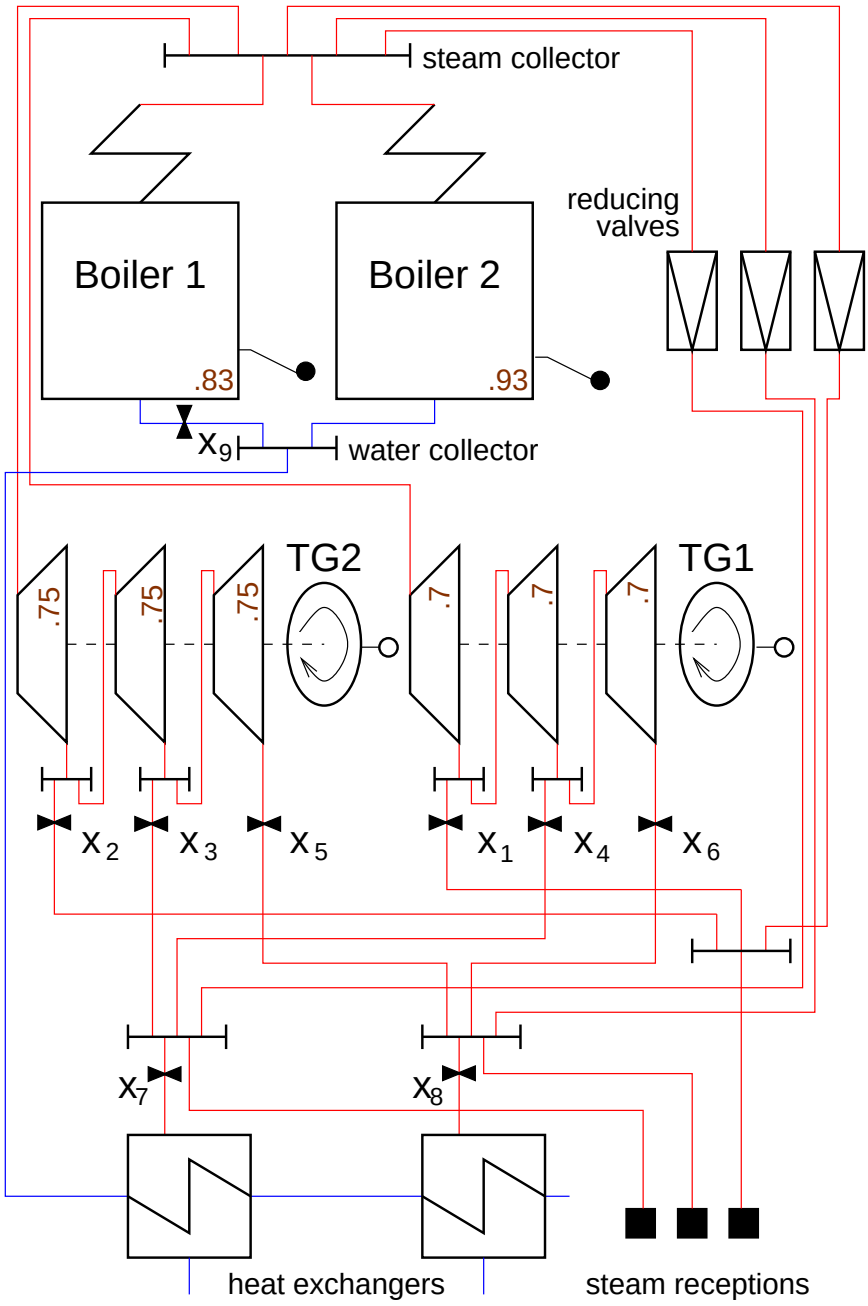

Figure 11: The diagram of test power system that is a simplified version of the plant model.

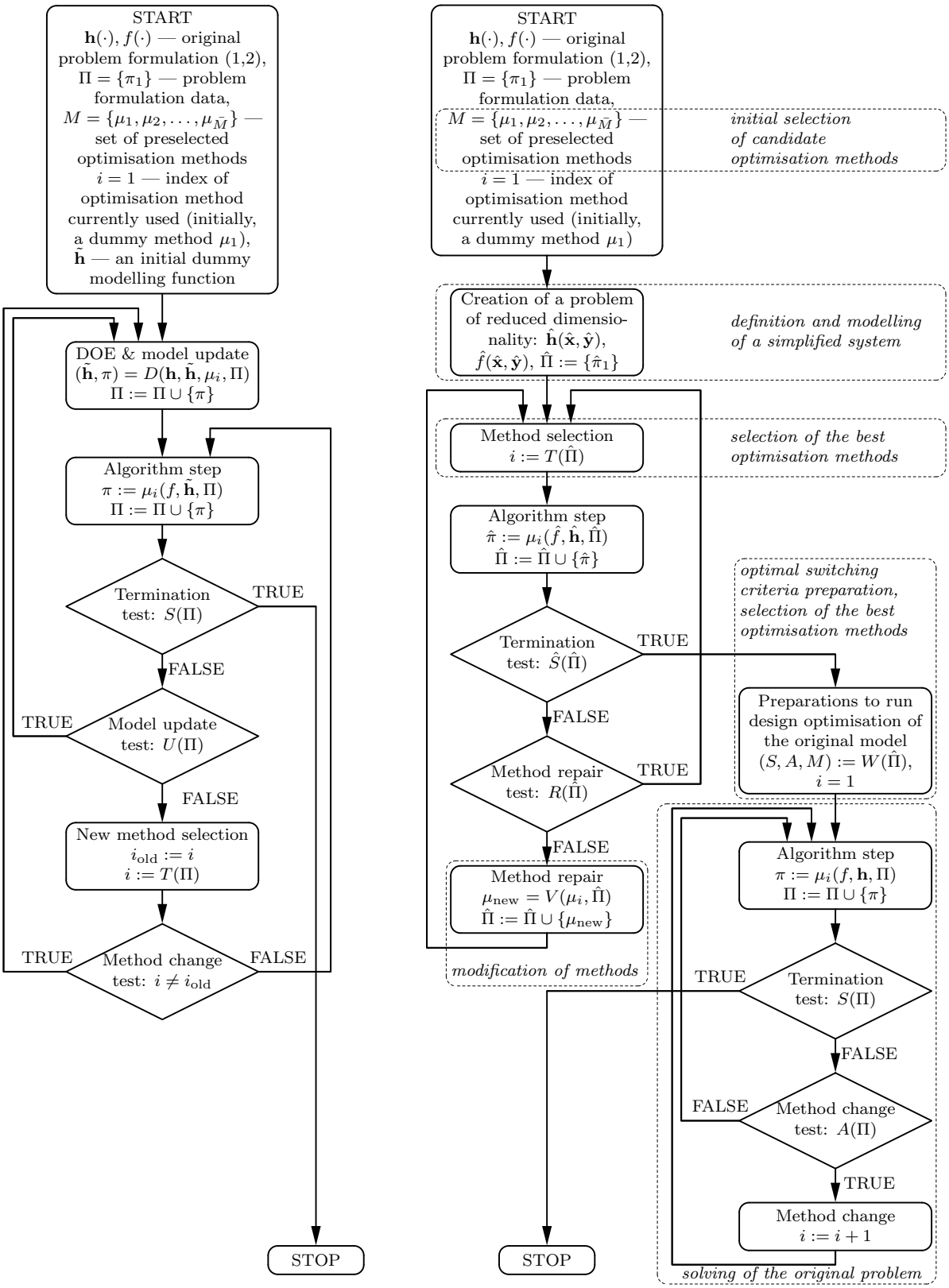

Figure 12: Flowcharts for the operation of the standard (left) and the proposed (right) approaches to design optimisation.