Int. J. Appl. Math. Comput. Sci., 2007, Vol. 17, No. 1, 53–71
DOI: 10.2478/v10006-007-0006-3

HYBRID APPROACH TO DESIGN OPTIMISATION: PRESERVE ACCURACY, REDUCE DIMENSIONALITY

MARIUSZ KAMOLA

NASK, ul. Wąwozowa 18, 02–796 Warsaw, Poland
e-mail: `Mariusz.Kamola@nask.pl`

Institute of Control and Computation Engineering, Warsaw University of Technology
ul. Nowowiejska 15/19, 00–665 Warsaw, Poland
e-mail: `M.Kamola@ia.pw.edu.pl`

The paper proposes a design procedure for the 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, a small set of simple optimisation methods is chosen, out of which those performing best will constitute the hybrid algorithm. The simplicity of the method allows 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 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 not having much knowledge of the system or modelling technology, but having the basic practice in optimisation. It is designed as a compromise between brute force optimisation and 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 the 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 can be calculated only by some complex algorithm. Here, we call an algorithm complex when its overall operation cannot be represented by analytic formulae. This means that the algorithm output is unpredictable.

There may be various reasons behind such complexity. The first one is the computation accuracy, which may make even simple procedures (like square root computation) "complex" if the data representation precision is inadequate. However, sheer increasing of the precision may not help if the algorithm itself is extremely sensitive to disturbances (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 a large number of cases the algorithm introduces (artificial) randomness reflecting the (true) nature of some real object—the randomness the optimisation procedure has to deal with.

Naturally, the algorithms considered model the behaviour of various systems. They are the mathematical models of those systems in different phases of their life. Accordingly, 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 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 were once described accurately by Sandia Laboratories (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 the one hand, unpleasant properties of the objective and constraints tempt to use brute force, i.e. some very unrefined routine, often executed massively in parallel. On the other hand, every function evaluation requires lengthy computations to be run. This implies that the selected optimisation routine should be efficient, effective and robust. The methodology of such a 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 not be known whether the unpleasant model properties are just a result of an inaccuracy or are inherent to the modelled system. In plain words, there may be no "knob" for adjusting the modelling accuracy—and the optimisation routine must take the numerical model as it 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 (without even making any changes). It is frequently described as a "black box"—with the difference that a true black box never explodes, while a third-party simulator—sometimes does.[1]

[1] Drawing such an analogy is completely adequate here: possible malicious behaviour of simulators includes hanging or casting an exception that may destroy the controlling optimisation module.

1.1. Problem Formulation.

The design optimisation problem can be formally defined as follows:

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

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

$$\begin{cases} h_1(\boldsymbol{x}, \boldsymbol{y}) &= 0, \\ \quad\vdots & \\ h_N(\boldsymbol{x}, \boldsymbol{y}) &= 0, \end{cases} \tag{2}$$

or, briefly, $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0}$ is satisfied. Here, $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y})$ represents relationships between the input and output of the modelled system. Usually, the optimisation process is additionally constrained by

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

The domains $D_{\boldsymbol{x}}$ and $D_{\boldsymbol{y}}$ 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 and 387).

Therefore, the set of feasible decision variable values is determined by three restrictions: explicitly by (3a), implicitly by the existence of a 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 regarding the reason or degree of constraint violation.

A typical approach to solve design optimisation problems will be briefly presented in Section 2. Next, two practical problems of the kind considered will be presented in Section 3, and followed by the author's proposal of a methodology for solving them that is alternative to the typical approach. The results of applying such a 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 the system behaviour. The analysis is usually performed by simulation or a modelling module, but it can also be done in a real system. Such experiments are costly in the sense of both money and time consumption. The time is particularly inconvenient as it is an inevitable cost that cannot be substituted by anything else.

A widely recognised and applied strategy to overcome such a difficulty is to perform optimisation using a "surrogate model", i.e. some cost-effective but less accurate model to calculate the objective value. The remainder, i.e. the modelling and optimisation techniques, is just the effect of following the paradigm of simplified modelling. Let us discuss the three component strategies: how to choose points for the tuning surrogate model, how to form such a model, and how to find an optimum using such a 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 the system performance. The DOE is usually performed at the initial stage of design optimisation with the purpose of detecting 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, the factorial DOE and orthogonal arrays. In the case of random or quasi-random sampling, simulations are run for a number of feasible points with equal probabilities of being selected. In the 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 the 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 the construction of a surrogate or approximate model. Using such model responses for optimisation, instead of running the "true" experiments, is called the Response Surface Methodology (RSM). The term RSM nowadays means the application of simplified models, of which the polynomial, neural, Bayesian and Krigging ones are most popular. Originally, the RSM was fitted by linear, quadratic, cubic (and so on) regression 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 is considered to be a good compromise between the classical RSM and Bayesian regression.

Many design optimisation packages make it possible for the the user to specify a custom model template: using a model matching well a particular problem is worth more than any general purpose standard RSM technique. Such a 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. e.g. (Plambeck *et al.*, 1996). It should be emphasised that at 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 heavily depends on whether the gradients of the objective function and constraints with respect to the decision variables can be reliably estimated. If yes, the 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: the Nelder-Mead simplex search, simulated annealing, controlled simplex and complex searches of many kinds and, most of all, evolutionary strategies. The 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: the DOE, the 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 fit 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 the original modelling accuracy, it can model all output variables or only selected and adequate ones. This is strongly linked with the type of the optimisation routine, which may change with the progress of optimisation.

2.4. Present Design Optimisation Packages. Let us make a brief overview of commercial integrated design optimisation tools present on the market. The goal is to show that issues of interest in this paper, i.e. the support for simulation failures and flexibility in combining the DOE, the RSM and optimisation, are addressed with varying attention. The questions are as follows:

- Can the modelling be done by an external "black-box" module, or must it use the procedur-supplied modelling language?

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

Package name	“black-box” model supported	opt. routine supplied by the user	opt. method chosen adaptively	RSM model supplied by the 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 by the user.

- Can the optimisation be done by a 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 the RSM with a user-supplied model supported?
- Are simulation failures safely handled?

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

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

The key issue is how to infer about the model output distribution for a given design. This is typically done by producing a number of realisations of random input values in a more or less systematic way. Then a number of simulations are performed 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 for the optimisation accuracy.

3. Examplary Problems and the Proposed Methodology

The main innovation regarding the design optimisation proposed in this paper is the alternative way of constructing the optimisation algorithm. The suggested procedure comes as a 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 present in the community of power engineers for long time (Portacha, 1969). As the modelled systems consist of many elements, modelling them using polynomial models usually fails due to the lack of the appropriate amount or sort of data that could be used for tuning them. So, another approach has been taken that utilises the 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

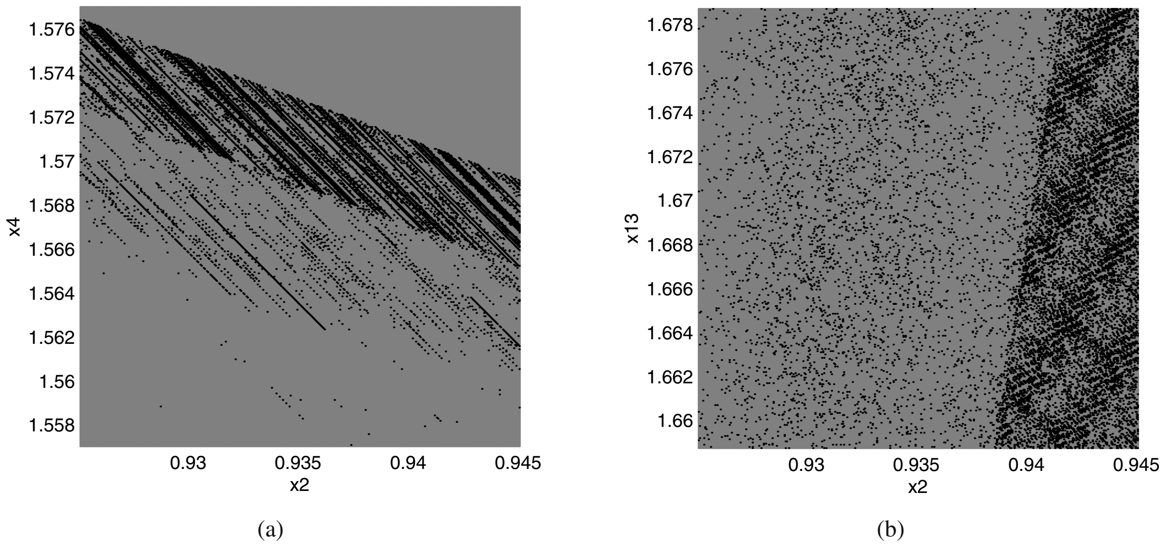


Fig. 1. Two-dimensional exemplary cuts through $D_{\boldsymbol{x}}$ with sets of decision variable values for which the modelling (i.e. computing $\boldsymbol{y}$ by simulation) fails marked in black.

of science and is extremely sophisticated but allows to model system behaviour already at the design stage.

The structure of the coal industrial power plant considered is comprised of boilers, turbines and regeneration blocks, with water circulating through them in cycles, see Appendix A for a 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 $\boldsymbol{y}$ being hidden inside the simulation module. The equation of the type (2) may be solved by finding, for a given $\boldsymbol{x}$,

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

using an optimisation procedure. Unfortunately, the solution to (4) satisfying (2) may not 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 functional blocks of the plant) that may be solved by the calculation of unknown variables by means of consecutive substitutions. The violation of any equation in the set is cancelled out by iterative adjustments of selected elements of $\boldsymbol{y}$. (Those elements play a role analogous to that which the vector $\boldsymbol{y}$ has in (4), but their number is lower; moreover, such an 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 $h_i(\boldsymbol{x}, \boldsymbol{y}) = 0$ in the set (2).

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

The performance index is the difference between the profit from selling electric energy and the cost 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 steam having 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 minimal cost appears several times a day, mainly because of the changing steam demands following the production schedule, which is usually known in advance. The optimal set-point location may also be affected by changes in the price of electric energy.

In the model, there are 21 decision variables, selected by a skilled engineer so that (2) is solved efficiently. They are subject to box constraints (3a). The main purpose of those constraints is to curb the range of decision variable values in an attempt to approximate roughly the set a trained human operator would consider reasonable for model input values.[2]

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

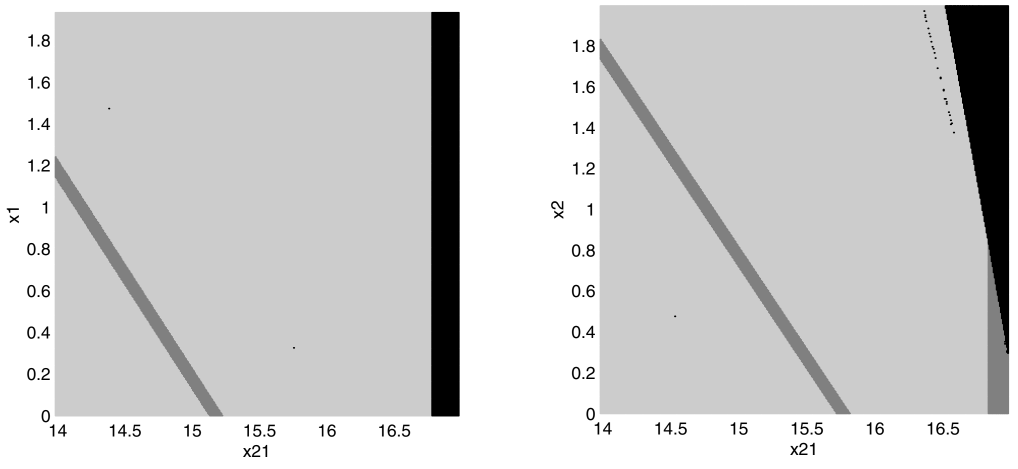


Fig. 2. Other cuts through $D_{\boldsymbol{x}}$ where feasible regions are drawn in light grey, regions with the implicit constraints (3b) violated drawn in dark grey, and regions with the implicit constraints (2) violated marked in black.

While inspecting the model behaviour, important adverse features of both constraints and objective function have been found that hinder the search for a solution. They were reported earlier in (Kamola and Malinowski, 2000). Regarding constraints, the implicit ones defined by the existence of a solution to (2) have very irregular shapes. Some of them are shown as black areas in Fig. 1, where exemplary cuts through the search domain $D_{\boldsymbol{x}}$ are drawn. They are in the 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 grey. It is interesting to point out that the areas of feasible points (light grey) can adjoin areas where the simulation fails altogether (the black colour means nonexistence of a solution to (2)). This is particularly inconvenient because the optimisation procedure may step from a "safe" area directly into a "minefield".[3] None of these problems matter too much when the design optimisation is to be performed by a user acquainted with the specifics of the modelled plant. In fact, the modelling module was made with the intention to be operated manually by such a user. While a specialist avoids dangerous or nonsense combinations of design variable values, an automatic optimisation routine wanders in every corner of $D_{\boldsymbol{x}}$, 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 nonroutine way it can discover completely new solutions.

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

The adverse features of the objective function mentioned above are the harshness and sudden steps of the response surface, as is shown on the directional graphs of $f(\cdot)$ in Figs. 3 and 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 $\boldsymbol{y}$ for given $\boldsymbol{x}$. The stepwise character is the effect of switching in the modelling formulae, which, for example, is the result of a transition of the working factor between fluid and gas states.

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. The analysis of the designed microwave circuit behaviour is performed nowadays by electromagnetic field simulators rather than by using field theory open formulae. Simulators make it possible to calculate the properties of microwave elements of fancy shapes.

The modelling is done using the finite difference time domain method (Taflove, 1995), where both space and time domains are discretised; next, discrete Maxwell equations are employed to compute an electromagnetic field distribution in every single cell of the partitioned object and in every single time instant. Therefore, the 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.

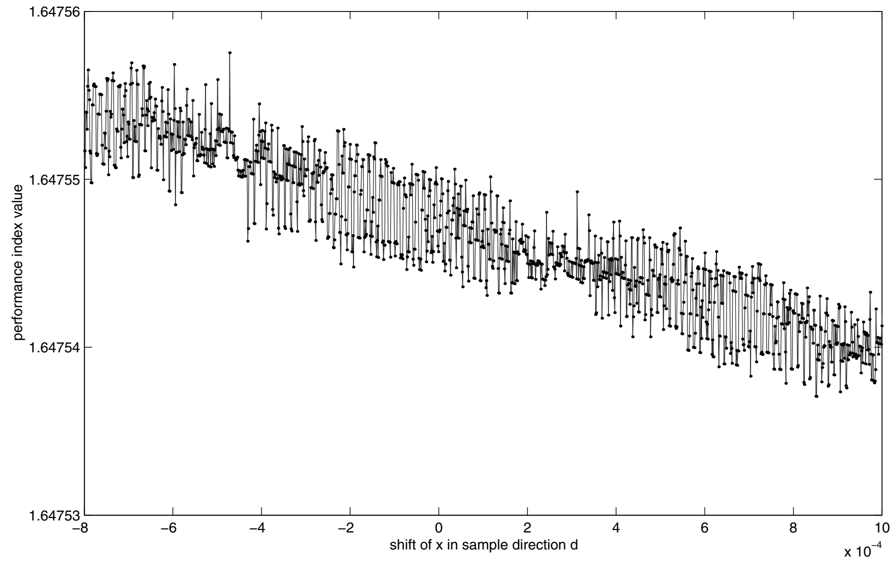


Fig. 3. Graph of the performance index computed close to the problem optimum along a given direction $\boldsymbol{d}$.

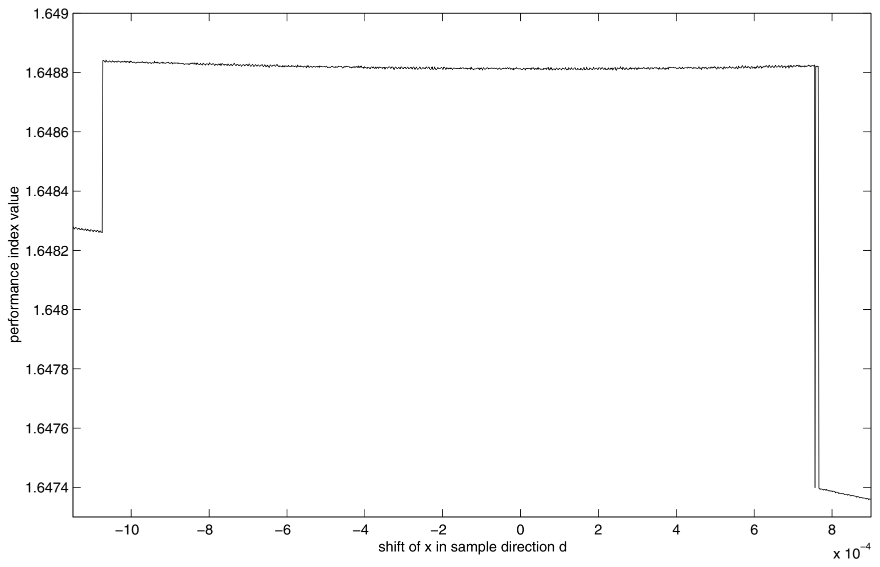


Fig. 4. Stepwise character of $f(\cdot)$ shown on the directional graph drawn near the problem solution.

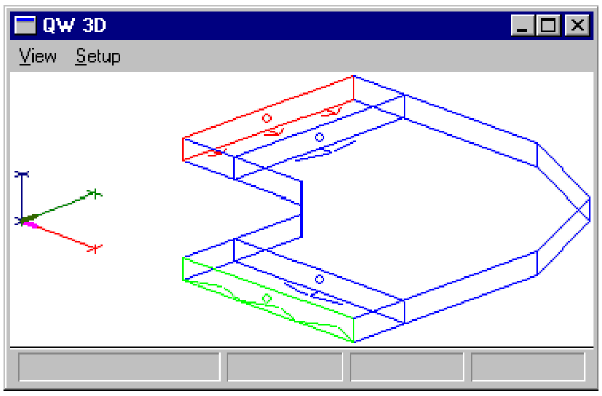
In the particular optimisation problem considered here (Kamola and Miazga, 2001), three dimensions x_1, x_2, x_3 of a waveguide bend are the decision variables, as is shown in Fig. 5. The waveguide is a kind of transmission line, frequently used in gigahertz frequency devices, and transmitting quite 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 a design in which the wave reflection coefficient of the waveguide remains 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 $\boldsymbol{y}$, then the performance index is

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

where $\boldsymbol{r}(\boldsymbol{y})$ is a linear penalty function, activated if the reflection coefficient exceeds some threshold. The penalty function $\boldsymbol{r}(\boldsymbol{y})$ computes a vector of penalties for all frequencies in the range considered, and $f(\boldsymbol{y})$ is the usual L_p norm of $\boldsymbol{r}(\boldsymbol{y})$. To complete the definition of the optimisation problem, $\boldsymbol{x}$ is subject to simple box constraints.

Unlike in the power plant problem, there is no way for the electromagnetic field simulator to fail, and so the implicit constraints (2) are always satisfied. Also, the



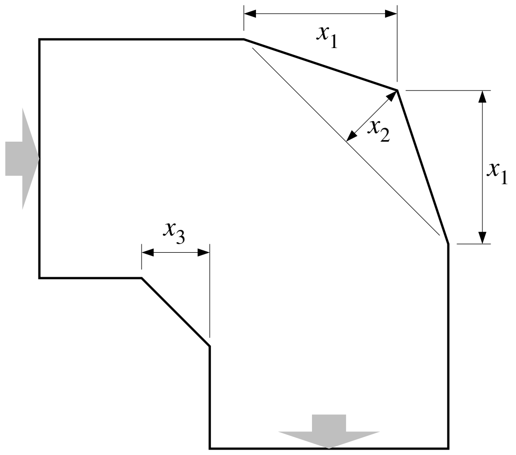


Fig. 5. Three-dimensional (left) and top (right) views of the waveguide bend subject to design optimisation.

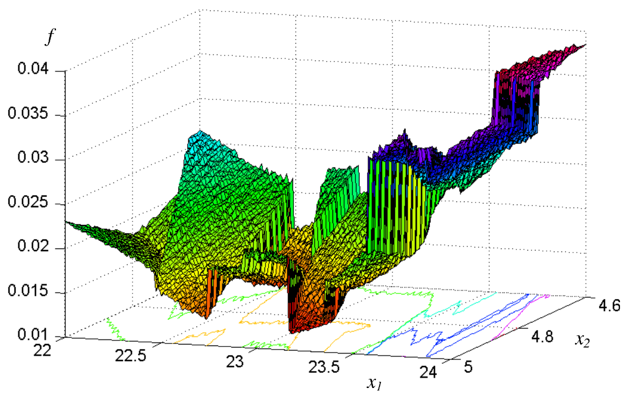


Fig. 6. Performance function for the waveguide bend design optimisation problem.

other sort of implicit constraints (3b) vanishes, incorporated into the penalty function $\boldsymbol{r}(\cdot)$. Therefore, the problem poses no difficulties related to constraints. However, the shape of the performance index itself is deteriorated 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 the 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 juncture an instant performance change is observed. The finer the discretisation, the larger the number of steps observed, their size decreasing.

3.3. Proposed Methodology for Solving Design Optimisation Problems. It has been shown that drawbacks of design optimisation problems may be rooted directly in the system nature, they may be 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) suggest that the models must first be verified and simplified before the optimisation process can start. However, it seems from the author's experience that in many cases changing the way the modelling is performed is impossible. This is either because the modelling routine is contained in an off-the-shelf analysis tool, or its intricacies are so serious that tracing errors and inaccuracies (e.g. by reverse engineering or in any other way) would take ages. Therefore, an optimisation engineer has his or her hands tied—at least as far as operation of the simulation module is concerned.

The admittedly, the designer cannot change the way the modelling is done, but he or she 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 or her own optimisation procedure, neither can he or she prescribe which optimisation routines built into the tool will be used in the course of optimisation, and in which order.[4] 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 a nongradient optimisation routine with a rough model, and local gradient optimisation routine with a fine one. But how can one know precisely which routine is most adequate at each stage of the optimisation process? The existing design optimisation packages require the user to use his or her own brains and problem knowledge to spec-

[4] Epogy (now iSight by Engineous, Inc.) is one of the very few exceptions to this rule; its operation and results will be used as a reference point in Section 4.

ify 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 extremely rudimentary. Therefore, the user does not know which optimisation routines to use, in which order, and what the termination criteria should be there, so that desirable solution quality 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 a microwave device manufacture) acquires modelling software, he or she expects his or her team to embed it into an optimisation routine quickly and effectively. Possibly none of those people knows much about the modelling tool (nor the optimisation theory) in order to be able to utilise this knowledge and to choose the optimisation routine matching best.

The methodology proposed in this paper tries to help the user with the construction of an 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 relatively small set of candidate optimisation routines capable to solve the problem jointly. They should be simple enough to allow some *ad hoc* modifications. If nothing is known about the problem, choose several direct search global methods to be run at 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 preserves the original system features. Use the original modelling tool and methodology to model it.
3. Use the selected optimisation routines to find a solution to the simplified problem, and to determine a 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 in the optimisation routines so that they are capable of solving this problem. If this does not help, look for some other optimisation algorithms. (The simplicity of the algorithms in the candidate set will make rapid modifications possible).
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 allows the engineer to inspect problem features while making the optimisation procedure, which—when applied to the original problem—should work properly without undue effort spent. Section 4 present the results of applying this methodology to the problems described in Sections 3.1 and in 3.2. Both the standard and proposed methodologies are presented in the form of flowcharts in Appendix 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. In general, those packages adopt much simpler models and are capable of modelling much simpler structures (e.g. turbosets not interlinked by common steam collectors, unlike ours). Consequently, they use simple, sometimes linear, optimisation methods. On the other hand, the modelling tool used employs highly nonlinear formulae. Eventually, two global procedures were qualified for preliminary optimisation and one gradient local procedure for the final optimisation stage.

Those global methods are a Controlled Random Search (CRS2) and an 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 high probability. The Generalised Reduced Gradient (GRG) method was chosen for final optimisation as it efficiently handles implicit constraints (2) without violating them in 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 the constraints (2) in the course of operation. By this choice, it was almost sure that the final hybrid algorithm would discover a 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 GRG.

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

Selection of the best optimisation methods. Numerous tests were performed with the purpose to find the best optimisation settings for CRS2 and the EA. They are reported in detail in (Kamola, 2004, pp. 87–94). The main conclusions are as follows:

- Discarding trial points for which the simulation fails does not make a difference for efficiency from the case of keeping them in the algorithm pool, marked with very high objective values. This applies to both CRS2 and the EA.
- Enhancing the algorithm's exploratory nature (by increasing the point pool size in CRS2 and by increasing the variability of mutations in the EA) comes at the cost of considerable 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 optimisation on its own as the optimal solution is situated in the active set for the constraint (3b), cf. Fig. 7, which is supported well by neither CRS2 nor the EA.

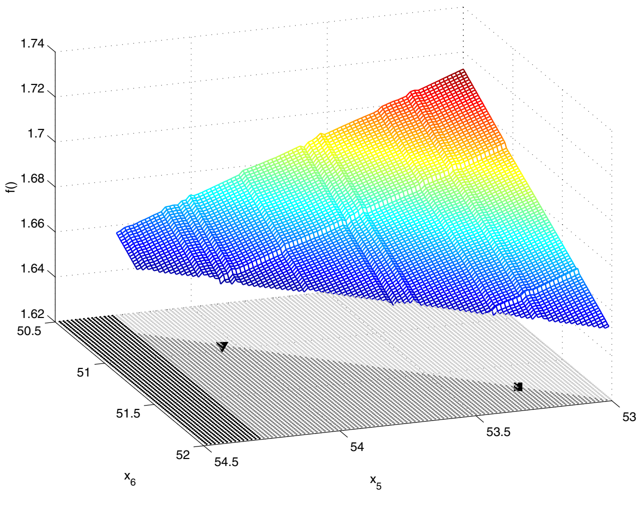


Fig. 7. Reference locations of the simplified problem solutions in the search subspace of x_5 and x_6 and the performance index surface. (The location of the CRS2 solution is marked with a triangle; the Epogy solution is marked with a square.)

- When run with cost-efficient settings (small pool point, little explorability), both CRS2 and the 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.

Table 2. Average performance index (boldface) values and the numbers of $f(\cdot)$ evaluations (italics) for simplified problem solving by CRS2 and the EA. The termination criterion was the relative accuracy ϵ_A (subsequent solutions differ by a ratio not exceeding ϵ_A).

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*

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

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 at computing to make objective gradient estimates using a finite forward difference scheme with adaptive step size selection revealed a great gradient variability (cf. Fig. 3). Consequently, cursory optimisation tests made with a simple sort of the steepest descent scheme completely failed, and the idea of GRG application had to be replaced by something else.

The 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 a 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 the case of COMPLEX, similarly to CRS2 and the EA, the optimal solution is sought by transformations of a pool of trial points, by means of the worst point reflection with respect to the reflection centre. In the case of an implicit constraint violation, a standard remedy in COMPLEX is to move the reflection centre toward the pool centre of gravity. The modification proposed by the author is based on the observation that such a gravity centre may be infeasible. But the best solution found so far *is* feasible, and thus it was made the reflection centre of last recourse, approached gradually when everything else failed.

Such changes made the procedure robust enough: the undesirable phenomenon of getting stuck before reaching a solution was eliminated as shown in Table 3, where the average performance indices and the corresponding numbers 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 moderate accuracy is required.

Table 3. Average performance index values (boldface) and the numbers of $f(\cdot)$ evaluations (italics) for the simplified problem solved by the COMPLEX routine without and with the author's improvements. (The parameter w is explained in the paragraph *Switching Criteria*.)

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

Switching criteria. Finally, the hybrid procedure consists of CRS2 being run with a pool size of $8(\dim \boldsymbol{x}+1)$, which is about the minimum pool size suggested in the literature. Then COMPLEX takes over, running with a pool size of only $2(\dim \boldsymbol{x} + 1)$ 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 significant variability in both the number of iterations and the solution quality, it was proposed that the switchover criterion be simple and not based on the current CRS2 efficiency. Consequently, the CRS2 operation was terminated after a given number of objective function evaluations, and COMPLEX was started from the best solution[5] found so far. The solution quality, the numbers of objective evaluations and the resulting efficiency are shown in Fig. 8. Also, the switchover after the first feasible solution was found is included as an extra criterion. It is evident, cf. Fig. 8(a), that increasing the CRS2 budget substantially has very little impact on the CRS2 solution quality, which is in turn even less correlated with the final solution quality, cf. Fig. 8(c). Consequently, with the budget growing, cf. Fig. 8(b), the total efficiency decreases, cf. Fig. 8(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 problem considered is 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 the 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 a parallel computation environment is available) would be the best strategy. The average results obtained for the simplified model are superior to those produced by a commercial design optimisation tool.[6]

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

Solving the original problem. The combination CRS2/COMPLEX was then applied to solve the original full-size problem. The only alterations in the algorithm parameters concerned the change in the point pool size, and setting the history window w to 120. It must be stressed that the algorithm finds the problem solution reliably and efficiently with no need for any further changes. The solution is reasonable in the opinion of specialists. This 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 effectiveness and efficiency of the author's methodology was possible at that stage.

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 the power plant problem. The replacement of the operator's experience and intuition with

[6] The objective function has the value of 1.646 at the best solution found by CRS2. In turn, the Epogy package found a solution with a performance value of 1.650. This value was found after nearly 25100 evaluations of $f(\cdot)$, and no further improvement was detected in the following 20000 evaluations. In much the same way as for CRS2, the Epogy solution is also located on the brink of the implicit constraint area (cf. Fig. 7). Nevertheless, the Epogy professional optimisation solver was apparently unable to 'glide' along it in pursuit of a better point. This confirms 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.

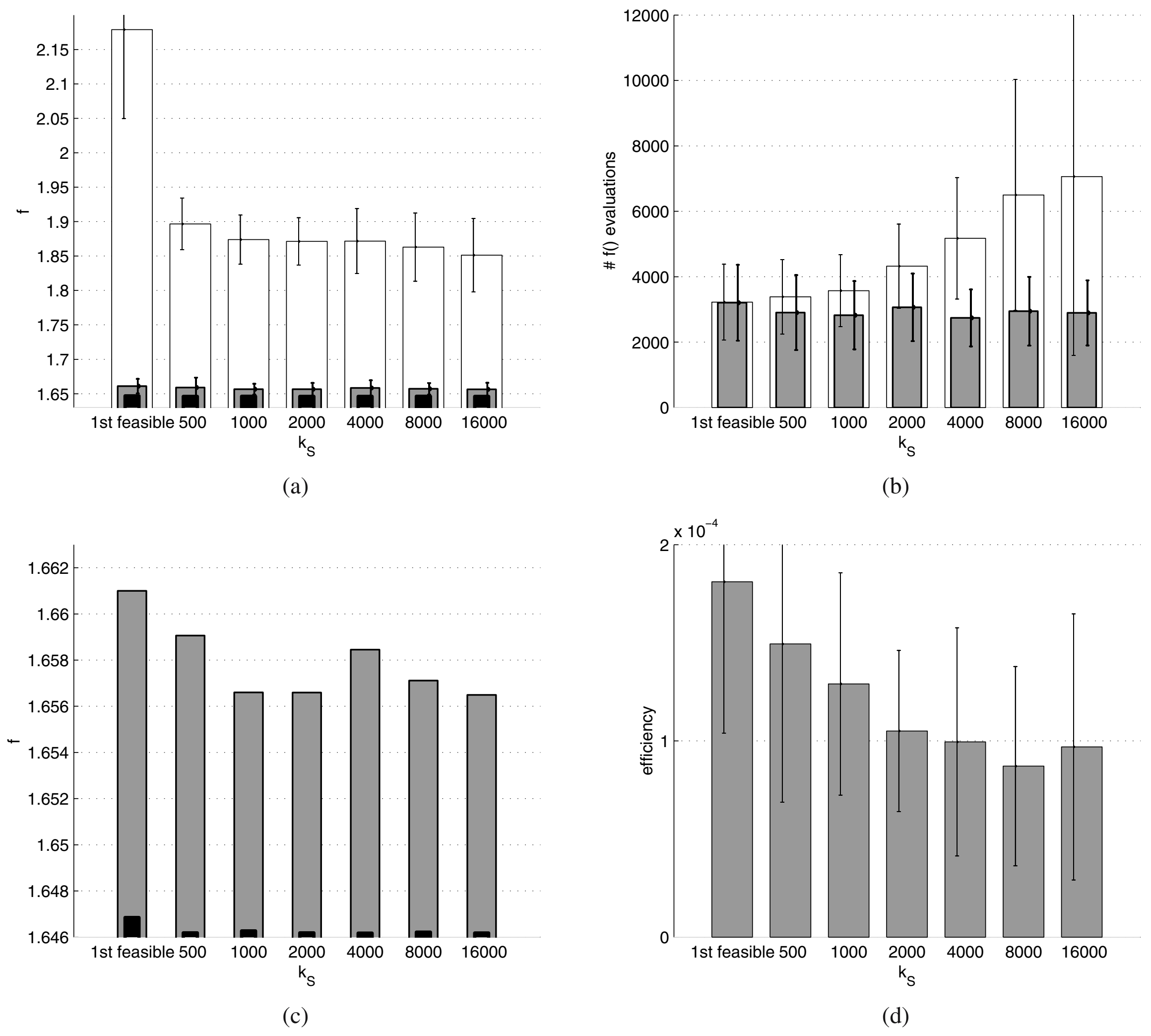


Fig. 8. Selected statistics for the operation of the CRS2/COMPLEX hybrid optimisation algorithm. Bars indicate average values; lines attached to them indicate standard deviations from the average values. Results are drawn for a varying switching criterion k_{S} being the number of $f(\cdot)$ evaluations after which a switchover from CRS2 to COMPLEX is performed. (a) The performance index value at the solution found by CRS2 (white bars) and COMPLEX (grey bars) along with the best results in a series of algorithm runs (black bars), (b) the total number of $f(\cdot)$ evaluations for COMPLEX (grey bars) and for the whole hybrid algorithm (white bars), (c) the complement of (a), (d) optimisation efficiency (an improvement in the solution quality divided by the number of $f(\cdot)$ evaluations) for the whole algorithm.

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

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

Selection of the best optimisation methods. In the case of the waveguide design there was no possibility to freely

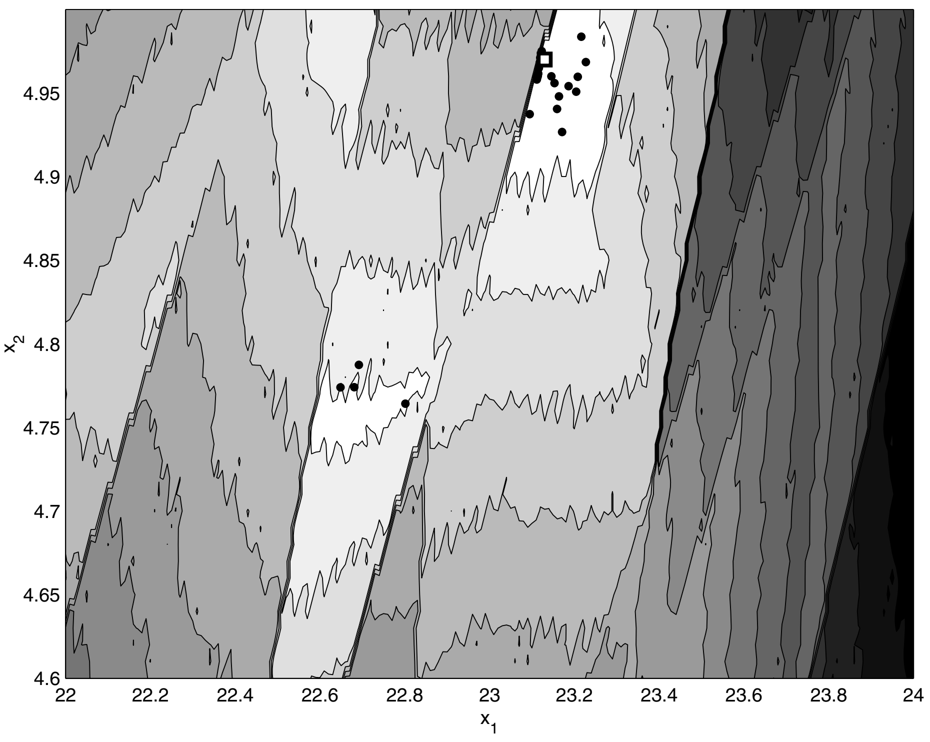


Fig. 9. Location of the CRS2 (black dots) and Powell (white square) solutions for the simplified waveguide design problem on the contour plot of the performance index. White areas denote regions of good performance (low values of the performance index).

test 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 operation of the optimisation routine were limited.

Separate tests of CRS2 and Powell's routine revealed that CRS2 needed more evaluations in a single run than Powell's method, but it found satisfactory solutions much more reliably. On the average, CRS2 run alone needs 30% less of the computational budget to work equally efficiently as Powell's method. This result seemed satisfactory, and CRS2 was assigned to do the whole work alone. Therefore, the parts *Reselection of methods* and *Switching criteria* of the proposed procedure were no longer applicable. Figure 9 presents the locations of several CRS2 solutions and of the best Powell solution. CRS2 solutions are sometimes located in proximity to a local minimum, but prevalently in proximity to the global one. Powell's 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 the constraints (3a), it was no longer cast on the search domain but "bounced" at the domain boundary back into the feasible region. This prevented the 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 consecutive worst points in the pool producing simultaneously as many trial points as is the number of active threads. Such a solution promotes explorability, rather than efficiency, but the so-modified CRS2 was found to work quickly and reliably enough.

Solving the original problem. To verify quite peculiar observations and assumptions about CRS2 capabilities, a number of optimisation trials were performed for the original 3-D problem. The 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 the performance index value reduced to values below 0.012, was 8 for Powell's method and 49 for the CRS2 routine. Respectively, the average numbers of $f(\cdot)$ evaluations after which the algorithms stopped were 167 and 1000.[7] Let us calculate the num-

[7] Alternatively, one could use the actual number of $f(\cdot)$ evaluations after which the solution was found. But this number is known

ber N of objective function evaluations needed to find a satisfactory solution with the probability p:

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

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. A smaller value of N means a more efficient method. Using (6) as the efficiency measure, we find Powell's method ($N = 6000$) still less efficient than CRS2 ($N = 4449$) in the 3-D case.

However, an increase in the required computational budget (due to the increase in the problem dimensionality) made it possible to reconsider the application of a hybrid CRS2/Powell routine. Fifty optimisation runs of such a routine were made, where Powell was activated after $k_S = 130$ or after $k_S = 260$ function evaluations made by CRS2. The results are given in Table 4, along with the

Table 4. Statistics for the 3-D waveguide design problem solved by different algorithms.

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

reference performance of CRS2 and Powell's method 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.

5. 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 the system considered, i.e. simpler than the original model (especially in terms of dimensionality) but preserving the features of the original system that are important for the optimisation process (the character of the constraints and of the response surface, etc.).
3. Preferring simple optimisation algorithms pays when unexpected and adverse problem features appear, as it is much easier to construct ad-hoc workarounds.
4. A recipe for making an efficient and robust optimisation routine can roughly be presented in the form of some design procedure; making the simplified model is both engineering and art, which is not subject to algorithmisation.

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

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 an in-depth problem analysis nor does it so by the application of the brute force—it leads another way in between those extreme approaches, somewhere besides the current standard approach to design optimisation problems. Additionally, it gives an engineer 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.

only when the stopping criterion is satisfied. The stopping criterion was set arbitrarily by a trained operator, and reflects both his knowledge of the behaviour Powell's method and disbelief regarding CRS2 applicability.

Acknowledgments

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

The experiments with the microwave bend optimisation example were carried out using the 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, Ph.D. (Institute of Radioelectronics, Warsaw University of Technology) for all the help and expertise.

References

Bazaraa M.S., Sherali H.D. and Shetty C.M. (1993): *Nonlinear Programming. Theory and Algorithms*. — New York: Wiley.

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

Bujalski W. (2001): *A Metod of Load Distribution in Power Systems Operatet through DCS*. — Ph.D. thesis, Warsaw University of Technology, (in Polish).

Edgar T.F. and Himmelblau D.M. (1988): *Optimization of Chemical Process*. — New York: 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*. — Tech. Rep. SAND2001-3515, Sandia Laboratories, available at `http://endo.sandia.gov/DAKOTA/papers/Reference3.3.pdf`

Hammel U. (1997): *Simulation models*, In: Handbook of Evolutionary Computation (T. Bäck, D.B. Fogel and Z. Michalewicz, Eds.). — Bristol: Institute of Physics Publishing, and Oxford, New York: Oxford University Press, pp. F1.8:1–F1.8:9.

Hyperworks (2006): Altair Hyperworks: Web page `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): *Mathematical model of a turboset working in changed conditions*. — Bulletin Inst. Heat Engineering, Warsaw University of Technology, 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*. — Proc. IFAC Sympos. Manufacturing, Modeling, Management and Control, MIM, Rio Patras, Greece, pp. 377–382.

Kamola M. and Miazga P. (2001): *Global and local optimization algotithms in automated wavequide design*. — Proc. 5-th Nat. Conf. Evolutionary Algorithms and Global Optimization, Jastrzębia Góra, Poland, pp. 97–105.

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

Papalambros P.Y. (1988): *Principles of Optimal Design*. — Cambridge: Cambridge University Press.

Plambeck E.L., Fu B.R., Robinson S.M. and Suri R. (1996): *Sample-path optimization of convex stochastic performance methods*. — Math. Program., Vol. 75, No. 2, 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 at `http://www.fe-net.org/technology/pso/`

Portacha J. (1969): *Optimisation of Heat System Structure in a Steam Power Plant*. — Ph.D. thesis, Warsaw University of Technology, Warsaw, (in Polish).

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

Price W.L. (1987): *Global Optimization Algorithms for a CAD Workstation*. — J. Optim. Theory Applic., Vol. 55, No. 1, pp. 133–146.

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

QWE (2003b): *QW-Optimizer User's Manual*. — Warsaw: QWED Ltd.

Sandia (2006): Features of computationally complex engineering problems. Web page `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*, Tech. Rep., Georgia Tech., available at `http://www.phoenix-int.com/library/papers.php`

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

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

Appendices

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 the 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 is 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 changing power demand or to failures, for example.

The diagram of the plant considered is given in Fig. 10. The efficiencies of selected devices are given in italics. Flows that are decision variables are marked with the symbol ⧗, with the decision variable index given beside. Industrial steam outlets are marked with the symbol ■. Flows or power levels contributing to the positive $f(\cdot)$ component in (7) are marked with the symbol •, and those contributing negative component—with the symbol ○. The factory needs three types of steam of a given pressure, temperature and flow, and one of a 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 a simple balance of the cost of coal consumption, the cost of pump operation and the gains from selling the electric power:

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

where $\boldsymbol{y}_{\mathrm{C}}^T = [y_{\mathrm{C},1}\ y_{\mathrm{C},2}\ \dots]$ is the vector of coal flows (expressed in kg/sec), $\boldsymbol{y}_{\mathrm{P}}^T = [y_{\mathrm{P},1}\ y_{\mathrm{P},2}\ \dots]$ is the vector of power consumption by plant pumps (expressed in kilowatts), $\boldsymbol{y}_{\mathrm{E}}^T = [y_{\mathrm{E},1}\ y_{\mathrm{E},2}\ \dots]$ 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 the prepared reduced dimensionality plant model is presented in Fig. 11.

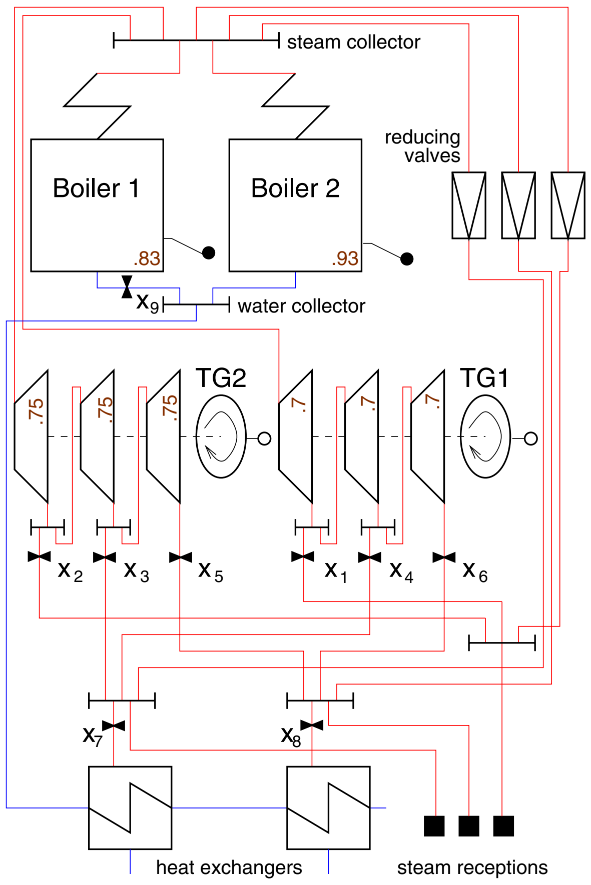


Fig. 11. Diagram of the test power system that is a simplified version of the plant model.

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, $\Pi = \{\pi_1, \pi_2, \dots, \pi_{\bar{\Pi}}\}$. 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 the location of solution approximations, the values of the performance index, and the information of the context in which it has been obtained (e.g. the index of the optimisation method used, the step number, etc.) ($\hat{\Pi}$ refers to the same kind of data for the model of reduced dimensionality.)

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

i – the index of the currently used optimisation method.

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

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.

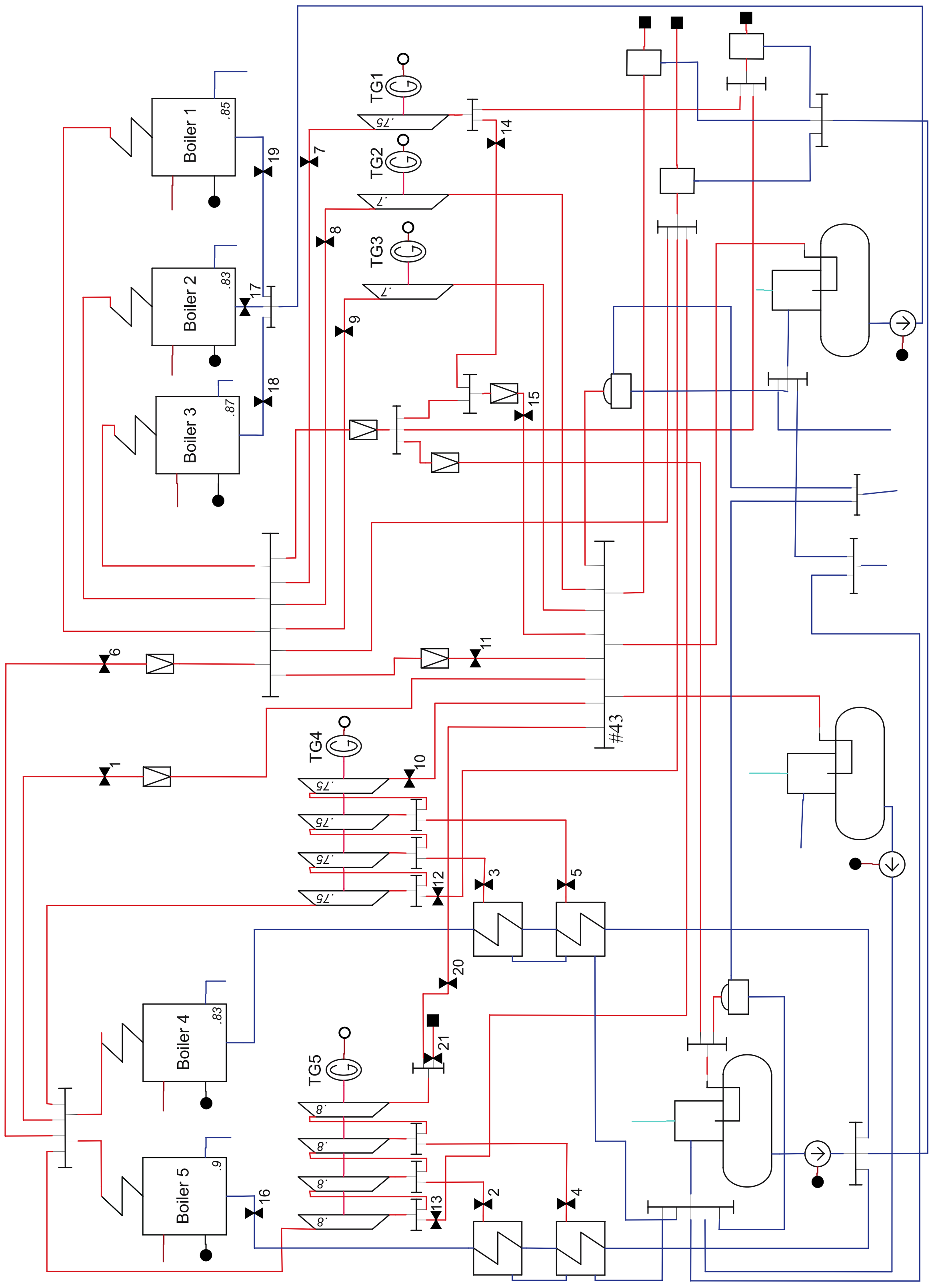


Fig. 10. Diagram of the industrial power plant at the ‘Janikosoda’ chemical works in Janikowo, Poland.

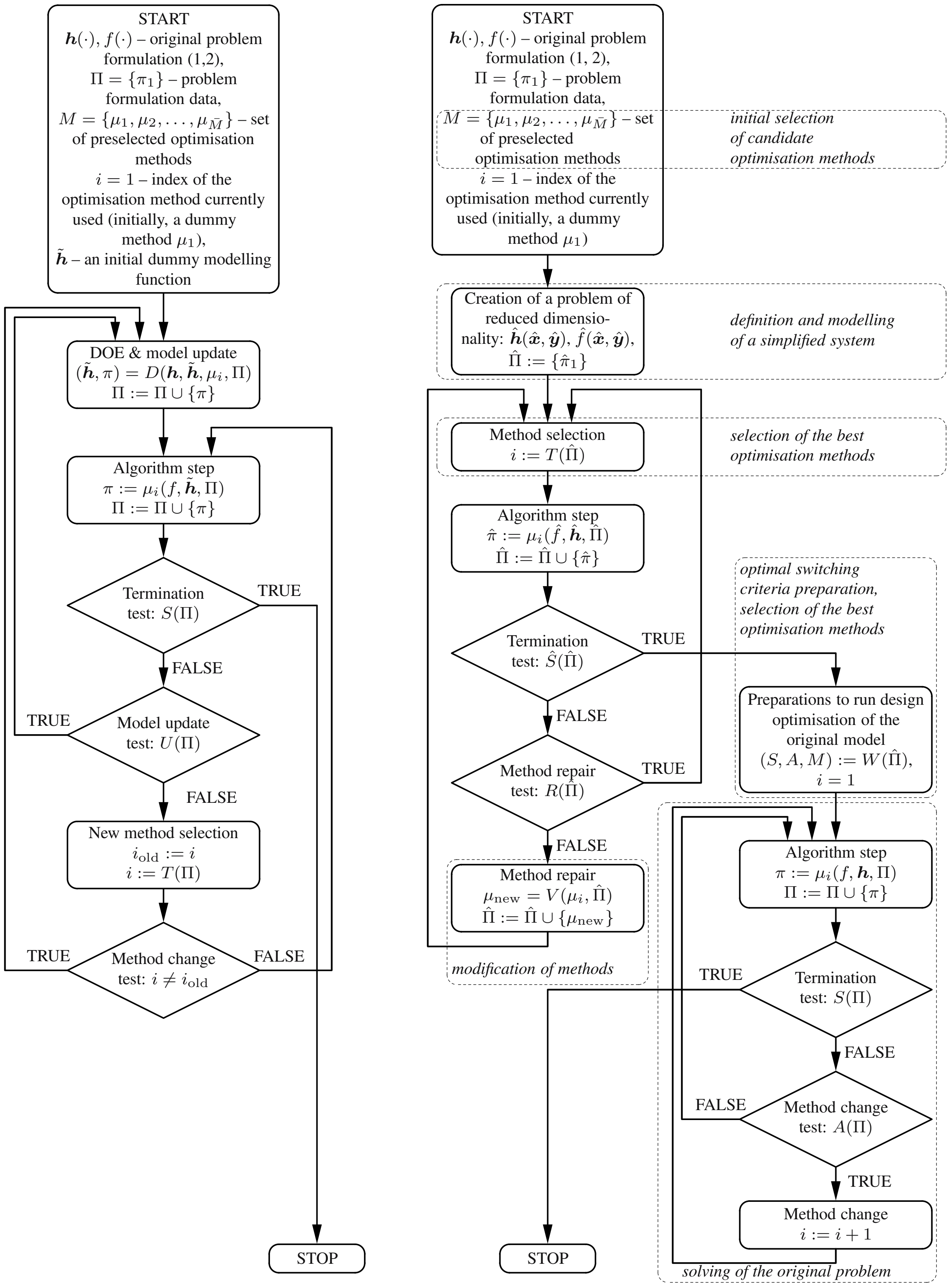

START
$\boldsymbol{h}(\cdot), f(\cdot)$ – original problem formulation (1,2),
$\Pi = \{\pi_1\}$ – problem formulation data,
$M = \{\mu_1, \mu_2, \dots, \mu_{\bar{M}}\}$ – set of preselected optimisation methods
$i = 1$ – index of the optimisation method currently used (initially, a dummy method μ_1),
$\tilde{\boldsymbol{h}}$ – an initial dummy modelling function
DOE & model update
$(\tilde{\boldsymbol{h}}, \pi) = D(\boldsymbol{h}, \tilde{\boldsymbol{h}}, \mu_i, \Pi)$
$\Pi := \Pi \cup \{\pi\}$
Algorithm step
$\pi := \mu_i(f, \tilde{\boldsymbol{h}}, \Pi)$
$\Pi := \Pi \cup \{\pi\}$
Termination test: $S(\Pi)$
TRUE
FALSE
Model update test: $U(\Pi)$
TRUE
FALSE
New method selection
$i_{\text{old}} := i$
$i := T(\Pi)$
Method change test: $i \neq i_{\text{old}}$
TRUE
FALSE
STOP
START
$\boldsymbol{h}(\cdot), f(\cdot)$ – original problem formulation (1, 2),
$\Pi = \{\pi_1\}$ – problem formulation data,
$M = \{\mu_1, \mu_2, \dots, \mu_{\bar{M}}\}$ – set of preselected optimisation methods
$i = 1$ – index of the optimisation method currently used (initially, a dummy method μ_1)
initial selection of candidate optimisation methods
Creation of a problem of reduced dimensionality: $\hat{\boldsymbol{h}}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}), \hat{f}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$,
$\hat{\Pi} := \{\hat{\pi}_1\}$
definition and modelling of a simplified system
Method selection
$i := T(\hat{\Pi})$
selection of the best optimisation methods
Algorithm step
$\hat{\pi} := \mu_i(\hat{f}, \hat{\boldsymbol{h}}, \hat{\Pi})$
$\hat{\Pi} := \hat{\Pi} \cup \{\hat{\pi}\}$
Termination test: $\hat{S}(\hat{\Pi})$
TRUE
FALSE
Method repair test: $R(\hat{\Pi})$
TRUE
FALSE
Method repair
$\mu_{\text{new}} = V(\mu_i, \hat{\Pi})$
$\hat{\Pi} := \hat{\Pi} \cup \{\mu_{\text{new}}\}$
modification of methods
optimal switching criteria preparation, selection of the best optimisation methods
Preparations to run design optimisation of the original model
$(S, A, M) := W(\hat{\Pi})$,
$i = 1$
Algorithm step
$\pi := \mu_i(f, \boldsymbol{h}, \Pi)$
$\Pi := \Pi \cup \{\pi\}$
Termination test: $S(\Pi)$
TRUE
FALSE
Method change test: $A(\Pi)$
FALSE
TRUE
Method change
$i := i + 1$
solving of the original problem
STOP
STOP

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

Symbols specific for the standard design optimisation approach:

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

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

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

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

R – the method repair test function, $R(\hat{\Pi})$. It is 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 the optimal hybrid algorithm preparation phase. The decision is, in theory, dependent on the whole information $\hat{\Pi}$ gathered so far.

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

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

A – a method change test function, $A(\Pi)$. It is used to determine whether the next optimisation method from M can be activated.

Received: 21 February 2006
Revised: 20 November 2006