SIMULATOR-OPTIMIZER APPROACH TO PLANNING OF PLANT OPERATION; ILL-DEFINED SIMULATOR CASE

Mariusz Kamola [*] **Krzysztof Malinowski** [*]

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

Abstract: In this paper an optimization problem is considered in which for every function evaluation a specialized simulation software must be run. This software computes the values of implicit variables. All variables are subject to constraints; additionally, for some trial points, the simulation fails to find corresponding implicit variables values. Adaptations of standard algorithms for this problem are suggested. Two hybrid algorithms are proposed. All the algorithms have been applied for steady state optimization of an industrial power plant model.

Keywords: Genetic algorithms, Global optimization, Power station control, Random searches, Simulation

1. INTRODUCTION

In many fields there exist needs to optimize design or operation of complex systems. Examples of such cases can be numerous, e.g. aeroplane wing design, drainage structure design or power plant operation planning, to mention some. To deal with them one must take advantage of pre-existing models that are the fruits of long-term experience and effort of the specialists in a given domain. These models are very often so complex that they cannot be presented in analytical form but take the shape of software codes that perform certain simulations to derive all model parameter values from a set of those specified by a designer.

To embed such a model in an optimization routine can become a demanding task. Spawning the simulation process is usually the essential part (also in terms of computational effort) of the objective function calculation, and there is no guarantee that it will be completed successfully. It is because the simulator itself was initially developed for manual use only and, apart from minor software interface incompatibilities, its behavior, namely its casual crashes, was no wonder for an expert and could be explained reasonably. On the contrary for optimization routine designer: being unable to compute either the function value or any of model parameters that would give the idea how far the considered decisions are from the domain can become a true problem.

Let us describe the optimization problem formally. We are to find the minimum of the objective function

$$\min_{\mathbf{x}} q(\mathbf{x}, \mathbf{d}) \tag{1}$$

with respect to the set (vector) of independent variables $\mathbf{x}$. The function depends also on vector $\mathbf{d}$ of implicit variables. The values of elements of $\mathbf{d}$ are computed, given the values of $\mathbf{x}$, by a simulation routine. Let us introduce a function

$$f(\mathbf{x}, \mathbf{d}) = 0 \tag{2}$$

which denotes this relationship. Additionally, let us introduce simple constraints on all variables:

$$x_i \in \langle l_i, u_i \rangle, \quad i = 1..n, \quad n = \dim \mathbf{x}, \tag{3}$$

$$d_j \in \langle l_j, u_j \rangle, \quad j = 1..m, \quad m = \dim \mathbf{d}. \qquad (4)$$

Of course, constraints (4) projected on the search space can turn into extra constraints of complicated nature. Moreover, for a certain value of $\mathbf{x}$ the simulation routine may not provide a solution of equation 2, constraining the optimization domain even more.

Given a problem of the nature as above one has to develop appropriate optimization approach based on properly adapted, modified, existing routines.

2. ALGORITHMS USED

It seems that there is no single algorithm that would suit the above optimization problem perfectly. All gradient routines can be fairly excluded since the objective function gradient is unavailable. Several of non-gradient ones have been chosen for tests. Let us present them shortly in their original shape, before they become subject to modifications to fit problem specific features.

2.1 *CRS2 Algorithm*

The abbreviation CRS2 comes from *Controlled Random Search, ver. 2*; the method was presented in (Price, 1987). It is a routine for global optimization that performs direct search, requires minimum preparation of a problem, and is applicable to constrained as well as to unconstrained optimization. The algorithm is summarized below (minimum of an objective function q is sought):

Step 1. Choose N points ($N \gg n$ where n is the problem dimension) at random over the domain V; evaluate value of the objective function at each point. The points chosen constitute the current point set.
Step 2. Find in the current point set a point $\mathbf{x}_l$ with the lowest value of q, and a point $\mathbf{x}_h$ with the highest value of q.
Step 3. Make a $n+1$-dimensional simplex using points from the current point set. The simplex must contain $\mathbf{x}_l$; the remaining points are chosen at random from the current point set. Compute a trial point $\mathbf{x}_t$ as the result of reflection of $\mathbf{x}_h$, the reflection center being the center of the simplex.
Step 4. If $\mathbf{x}_t \in V$ then evaluate $q(\mathbf{x}_t)$ and go to step 5. Else go to step 3.
Step 5. If $q(\mathbf{x}_t) < q(\mathbf{x}_h)$ then replace $\mathbf{x}_h$ in the current point set with $\mathbf{x}_t$ and go to step 6. Else go to step 3.
Step 6. Stop if the stopping criterion is satisfied. Else go to step 2.

CRS2 routine is not perfectly suited for our problem because evaluation of q in step 5 implies a simulation which can turn out to be unsuccessful. Also the case where (4) is not satisfied must be treated properly. Thus, modifications for these two cases have to be developed.

2.2 *Evolutionary Algorithm*

Various evolutionary algorithms stem from principles of evolution, from the idea of maintaining a population of individuals and letting them to evolve so that their "fitness" to a certain environment increases. A summary of experience in the field can be found in (Bäck *et al.*, 1997); it gives an outline for a specific routine as below (minimum of q is sought):

Step 1. Choose N points ($N > n$, n is the problem dimension) at random over the domain V. The points chosen constitute the initial population P_n, for the first time $n = 0$. Evaluate value of $q(\mathbf{x}_k)$ for every $\mathbf{x}_k \in P_n$.
Step 2. Find the best point in P_n and record the best point found so far. Check stop criterion, i.e. improvement of objective function value for best points found in several last steps.
Step 3. Create new population P_{n+1}. Each point is either the winner of a tournament (with probability p) or the result of a crossover of t tournament winners.
Tournament. Choose at random $\mathbf{x}_a$, $\mathbf{x}_b$ from P_n. Put in P_{n+1} the one that has better objective function value.
Crossover. Perform tournament t times to obtain a set of crossover participants. Perform floating-point crossover, i.e. compute centroid of tournament winners. Put the centroid into P_{n+1}.
Step 4. For every point $\mathbf{x}$ in P_{n+1} perform mutation, i.e. perturb each element of $\mathbf{x}$ with random variable ξ_i of Cauchy distribution; probability density function for ξ_i is

$$\frac{1}{\pi} \frac{\sigma}{\sigma^2 + (x - x_i)^2}.$$

Step 5. Assign $P_n := P_{n+1}$ and go to step 2.

Evolutionary algorithms (EA) and CRS2 have some common features in the context of our problem. Little knowledge of q is needed, nevertheless some amendments must be done to support objective function calculation failures and violations of constraints (4).

2.3 *COMPLEX Algorithm*

This method (called originally *Constrained Simplex*) is an important upgrade of the standard simplex search algorithm. It eliminates the problem of maintaining simplex regularity as well as supports convex search domains. The generic algorithm, taken from (Box, 1965), is as follows:

Step 1. Given an initial feasible point $\mathbf{x}_0$, create a set C (referred to as complex) of k points, $k \geq n+1$. This is done by a series of k augmentations. Calculate the value of q for every complex element.

Augmentation. Calculate the center of C. Choose at random a trial point $\mathbf{x}_t$ from the space constrained by (3). Move $\mathbf{x}_t$ halfway towards C center so many times that (4) is satisfied. Then set $C := C \cap \{\mathbf{x}_t\}$.

Step 2. Stop if the stop criterion is satisfied. Else go to step 3.

Step 3. Mark the worst (in terms of q value) element of the current complex C as $\mathbf{x}_h$. Calculate the center $\mathbf{x}_c$ of $C \setminus \{\mathbf{x}_h\}$. Compute a trial point $\mathbf{x}_t = \mathbf{x}_c + \alpha(\mathbf{x}_c - \mathbf{x}_h)$. Set all elements of $\mathbf{x}$ that violate (3) on appropriate bounds.

Step 4. Check if $\mathbf{x}_t$ satisfies (4). If not, set $\mathbf{x}_t := 0.5(\mathbf{x}_c + \mathbf{x}_t)$ and repeat this step; else go to step 5.

Step 5. If $q(\mathbf{x}_t) < q(\mathbf{x}_h)$ then replace $\mathbf{x}_h$ in C with $\mathbf{x}_t$ and go to step 2. Else set $\mathbf{x}_t := 0.5(\mathbf{x}_c + \mathbf{x}_t)$ and go to step 4.

The above scheme, unlike the previous ones, takes into account the existence of implicit variables and related constraints (the stage of computing $\mathbf{d}$, given $\mathbf{x}$, in order to evaluate q has not been shown in the routines). However, feasibility requirement for an initial point $\mathbf{x}_0$ can be sometimes hard to fulfill. Also the assumption for domain convexity is much too strict for our problem. Unlike in the case of CRS2 and EA, q calculation failures can be considered in step 4 simply as the violations of constraints (4), and treated respectively.

3. ALGORITHMS ADAPTATIONS TO THE PROBLEM SPECIFICS

The real problem that had to be originally solved was a steady-state optimal working point computation for a model of a boiler-turbine system (Skowroński and Bujalski, 1999).

The model diagram is presented on figure 1. Two boilers supply common steam collector. The steam can be then distributed to the steam receptions via two three-stage turbines as well as through the set of reducing valves. Steam parameters (temperature, pressure, flow) at the steam reception points are well defined since the steam is directed from there for further use in factory installations. The goal is to find values for process variables so that the operation costs are minimal.

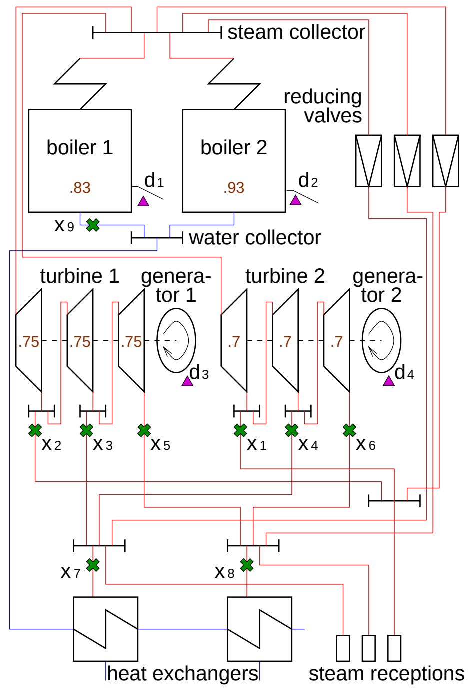


Fig. 1. Industrial power unit diagram

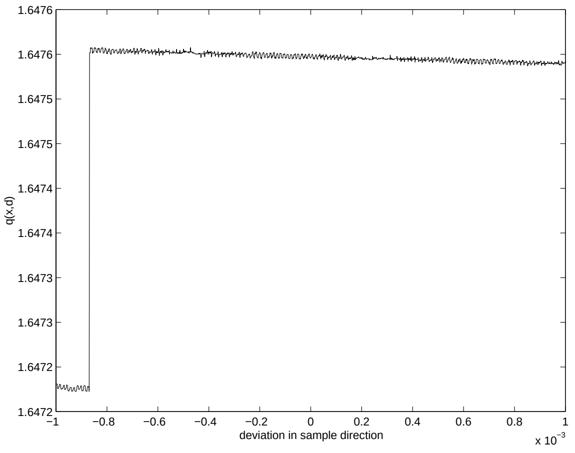
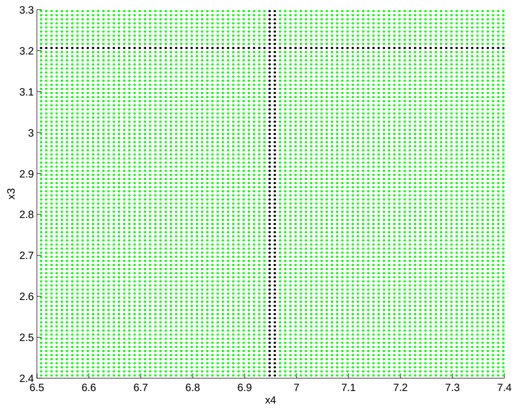


Fig. 2. Sample graph of $q(\mathbf{x}, \mathbf{d})$ for small changes of $\mathbf{x}$

The software dedicated for a class of simulation problems like this has been developed in The Institute of Heat Engineering, Warsaw University of Technology. The user is to select the set of independent variables and to determine their values. (In our case all explicit (decision) variables are flows at points marked with crosses.) The simulator computes all the remaining flows, entalpies, pressures, powers etc. (Those influencing running-costs are marked with triangles). The simulation software operates on a set of algebraic equations, some of them being highly nonlinear. Given the values of $\mathbf{x}$ it tries to eliminate by simple substitution as many implicit variables as possible. However, should nonlinear equations remain, they are solved iteratively.

The objective function is defined as follows:



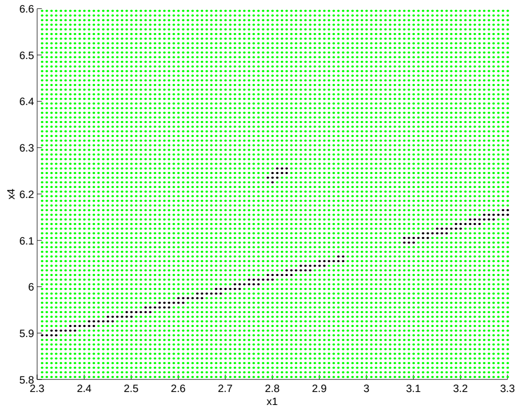


Fig. 3. Sample 2-dimensional sections through the domain

$$q(\mathbf{x}, \mathbf{d}) = 0.3 (d_1 + d_2) - 5.555 \cdot 10^{-5} (d_3 + d_4) \tag{5}$$

where d_1, d_2 — coal flows $[kg/s]$; d_3, d_4 — power levels of generators $[kW]$. The coefficients represent coal and electricity prices, respectively. The fact that function q (being itself rather a simple one) takes as arguments variables computed by simulation process has serious consequences. Let us look at figure 2. It represents changes of q while moving $\mathbf{x}$ in a sample direction. The decreasing trend is distorted by apparent random peaks of both signs; they are symptoms of simulation inaccuracies. Moreover, sudden steps can be revealed, like that one at -0.87 on abscissa. These, in turn, are results of simulator internal function switching.

Another difficulty raises from the fact that equation 2 cannot be solved for some $\mathbf{x}$ that satisfies (3). This imposes extra implicit constraints on the optimization domain. Figure 3 presents two sample sections of the domain. (Points that caused the simulator to fail are blackened.)

Therefore, the problem specifics require from optimization routines to handle two additional cases: equation 2 solving failure (i.e. simulation finishes abnormally) and violation of (4). The latter case has been already solved in COMPLEX. In CRS2 and in EA it can be treated in common way, namely by introducing the penalty function:

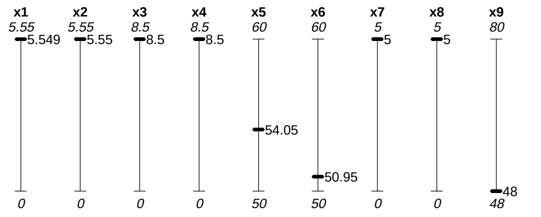


Fig. 5. Values of $\mathbf{x}$ in a sample solution

$$q_p(\mathbf{x}, \mathbf{d}) = q(\mathbf{x}, \mathbf{d}) + \sum_{j=1}^{m} p_i(d_j)$$

$$p_j(d_j) = \begin{cases} M(l_j - d_j) & \text{when } d_j < l_j \\ M(d_j - u_j) & \text{when } d_j > u_j \\ 0 & \text{else} \end{cases} \tag{6}$$

It is possible to introduce in both CRS2 and EA the same measures to handle simulation failure at $\mathbf{x}$:

- to assume that $q(\mathbf{x}) = \infty$, or
- to try computing another trial point $\mathbf{x}$, until simulation succeeds.

Two main disadvantages of COMPLEX algorithm are the requirement for initial feasible point $\mathbf{x}_0$, and the assumption that $\mathbf{x}_c$ is always feasible. As for the initial point, it must be delivered by an expert or by some preliminary less demanding algorithm (like CRS2 or EA). The second drawback can be overcome by an extra augmentation performed when $\mathbf{x}_c$ is infeasible. This solution was proposed in (Findeisen *et al.*, 1980), but in practice, however, it can turn out to be insufficient: gradual approaching an infeasible complex center does not guarantee that any feasible point will be found. One of possible improvements is to move artificially "complex center" (let us keep its name) halfway towards the best point in the complex. Eventually, one will be able to find a point $\mathbf{x}_t$ that satisfies (4).

The same rule can be applied when the reflection subroutine does not yield any $q(\mathbf{x}_t)$ better than $q(\mathbf{x}_h)$ or even reveals subsequent infeasible $\mathbf{x}_t$.

4. RESULTS OBTAINED

Each of the optimization algorithms with the proposed improvements was implemented and applied for the problem of optimal working point computation. Attempts were made to find the exact solution by running CRS2 or EA algorithms, although they were perceived rather as tools for coarse, preliminary optimization. Nevertheless, their simplicity as well as necessity to detect correct algorithm parameter values justify the approach.

CRS2 finds solutions that for experts seem to be satisfactory, but a single optimization requires

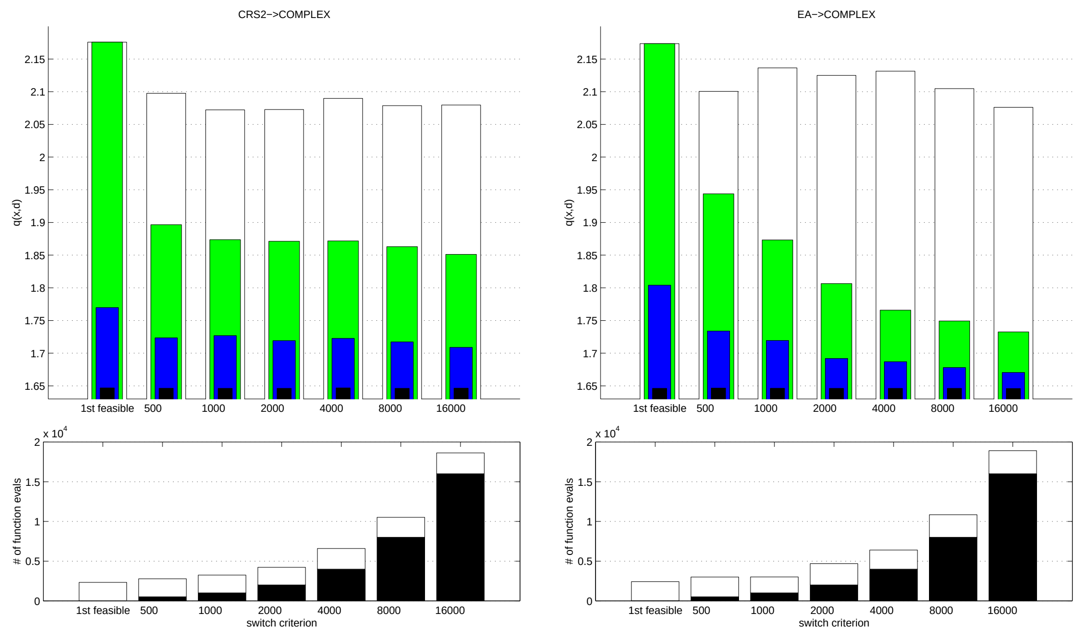


Fig. 4. Values of q at optimization milestones and number of q evaluations for various switch criteria

over 100,000 function evaluations — such a number is certainly inadmissible, since a single simulation takes 0.2 sec. on average. Nevertheless, the function value decreases from about 2.1 at the first feasible solution found to 1.646 at the minimum. The solution does not conflict with the common sense (see fig. 1 where efficiencies are printed in grey, see also fig. 5 where $\mathbf{x}$ values are put on rulers): x_9 is kept low, $x_5 > x_6$ which means that the total steam flow though the more efficient turbine is bigger. Zero flows through the reducing valves (not shown) indicate that all the steam, before it reaches steam reception points, is utilized for electricity generation. An extra note must be made on x_5 vs. x_6 — no further reduction of x_6 in favor of x_5 can be done due to implicit constraints violation.

The experiments have shown that setting $N = 8n$ for CRS2 was a reasonable compromise between algorithm speed and search profundity.

EA has turned out to be more efficient — about 40,000 evaluations of q are needed to achieve the value of 1.66 which is slightly worse than in case of CRS2. The tests against σ value in Cauchy distribution have shown that setting "the tail" either too fat or too thin badly influences EA convergence. (Taking σ so that $P\{\text{"mutated } \mathbf{x} \text{ satisfies (3)"}\} = 0.95$ turned out to be a good choice.) The value of σ should be modified during the algorithm execution, if EA were to be the target algorithm for our problem.

In general, both CRS2 and EA behaved better if for a simulation failure it was assumed that $q(\mathbf{x}) = \infty$.

Since COMPLEX requires an initial feasible point $\mathbf{x}_0$, it has been provided as one of intermediate points found in CRS2 operation. The tests were run to find out whether COMPLEX could be applied for refining CRS2 or EA results, therefore $\mathbf{x}_0$ was chosen so that $q(\mathbf{x}_0) = 1.6473$. In fact, the routine was able to improve the result easily, although the ultimate values of q varied from program run to run. (This conclusion conforms to those made in (Box, 1965).) The best value for k was the doubled problem dimension.

It can be observed that the nature of COMPLEX makes it an ideal algorithm in the final stage of optimization, a complement for either CRS2 or EA. Series of tests were run to discover characteristics of such hybrid algorithms for various switching criteria. The results are illustrated on figure 4.

Seven criteria have been tested: "1st feasible" means that COMPLEX is initiated immediately as CRS2 (or EA) finds the first feasible point. (This point becomes $\mathbf{x}_0$ for COMPLEX.) The other ones, i.e. "500", "1000" and so on, limit the maximum number of function evaluations done by CRS2 (or EA) to 500, 1000,..., respectively. Then COMPLEX is run with $\mathbf{x}_0$ set to the best point found so far.

Embedded bars on the figure indicate average value of q for the 1st feasible solution (white), for the solution at which switching to COMPLEX is done (light grey), and for the optimal solution (dark grey). The average was calculated for 90 algorithm runs. The deepest embedded, black, bar shows the best solution found in a series. The lower graphs show the mean of total function calls for each criterion. White parts of bars stand for the number of function calls done by COMPLEX algorithm.

It can be observed that the average quality of solutions found by COMPLEX depends clearly on the quality of the start point. The trend is particularly distinct for the first three situations, i.e. where the start point is relatively poor, although the best solutions (black bars) seem to not depend on the switching criterion.

This fact implies that start points for COMPLEX should be computed effectively. In this field EA wins over CRS2, which in turn seems to improve its solutions very slowly as the limit of function calls increases.

5. CONCLUSION

The values obtained from series of simulations can be interpreted in terms of mean results, and in terms of extreme results. The user is limited to the first approach when none computation can be done in parallel. In this case EA/COMPLEX sequence is recommended, especially when considerable number of function evaluations is allowed. EA is much more prone than CRS2 to converge steadily.

If a multiprocessing environment is available, then it is best to spawn multitude of COMPLEX algorithms from various available points $\mathbf{x}_0$. As one can observe in figure 4, the best final solution over a series of simulations virtually does not depend on the $\mathbf{x}_0$ quality. Sad but true, for problems like this sheer force is as for now the best approach: to start many algorithms in parallel, and to choose, at the end, the best among the solutions found.

No matter how unrefined this approach may seem, it is able to solve the problem, eventually. Moreover, unlike stated in (Skowroński and Bujalski, 1999), reducing computation time to about 2000 sec. makes EA/COMPLEX sequence applicable on-line, since forecasts of steam demands are available an hour in advance.

It is hoped that this approach will be applied in practice as soon as parameters of a real power plant being the object of interest are precisely identified.

6. REFERENCES

Bäck, T., Fogel, D. B. and Michalewicz, Z., Eds.) (1997). *Handbook of Evolutionary Computation.* Institute of Physics Publishing and Oxford University Press.

Box, M. J. (1965). A new method of constrained optimization and a comparison with other methods. *The Computer Journal* **8**(1), 42–52. www3.oup.co.uk.

Findeisen, W., J. Szymanowski and A. Wierzbicki (1980). *Teoria i metody obliczeniowe optymalizacji.* Państwowe Wydawnictwo Naukowe. In polish.

Price, W. L. (1987). Global optimization algorithms for a CAD workstation. *Journal of Optimization Theory and Applications* **55**(1), 133–146.

Skowroński, P. and W. Bujalski (1999). A method for automatic generation of linearized mathemetical model of energy and technological systems. *Archiwum energetyki* **XXVIII**(3-4), 19–31. In polish (english abstract available).