Raport
Instytutu Automatyki i Informatyki Stosowanej
Politechniki Warszawskiej

Simulation Optimization — A Survey of Methodologies

Mariusz Kamola

Raport[1] nr 03 – 18

grudzień 2003

[1]Raport wykonany w ramach projektu KBN nr 7 T11A 022 20

Politechnika Warszawska
Instytut Automatyki
i Informatyki Stosowanej
ul. Nowowiejska 15/19
00–665 Warszawa
tel.: (48) 22 – 6607397
fax: (48) 22 – 253719

Abstract

A review is made of algorithms that optimize performance of complex systems by using simulation to evaluate the performance value. Optimization problems are divided into classes of stochastic and deterministic optimization. The focus is set on difficult cases where both the response surface and the search domain are complicated. The conclusions can be drawn that direct search methods, supported with parallel processing, are most widely used for practical applications in both classes. The conclusions can also be made on growing demand for general purpose reliable optimization routines, and they formulate the proposition for the architecture of such routines.

Keywords: stochastic optimization, simulation optimization, constrained optimization

1 Introduction

The ever increasing power of contemporary computers tempts scientists to elaborate more and more exact and complex models of real-life phenomena and systems. Such modelling is often demanding, in terms of computational effort, due to two major factors. The first is the natural complexity: the load of mathematical and practical knowledge put in the construction of a model results in numerous formulas whose solution can be obtained only numerically. The second is the uncertainty (e.g. described by probability distribution functions) that propagates within the model rendering it intractable but through lengthy simulations and averaging.

Behaviour of a system represented by those models often depends on certain parameters — the design variables. Setting their values so that the performance of the system is, in a desired way, efficient is the major concern of designers. Employing optimization methods for this purpose seems obvious, especially for the same reason as mentioned earlier: the growing computing power encourages utilization of expensive optimization methods rendering hope that the solution would be globally optimal.

Embedding a simulator in an optimization scheme is one of the main techniques and is the subject of this paper. In this section, the reader will be presented with the formulation of the simulation optimization problem, and some criteria that affect the choice of an optimization routine. The existence or no of uncertainty is a matter of dividing the material into two subsequent sections. Stochastic problems are described in Section 2, and deterministic ones in Section 3. The concluding remarks are given in Section 4.

1.1 Problem formulation

The problem of optimization through embedded simulation can be written down in its general form as

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

where $f(\cdot)$ is some performance function of a vector of design variables $\mathbf{x}$. The value of $f(\cdot)$ is computed through simulation of a given system. The optimal design, denoted by $\mathbf{x}^\star$, is sought that minimizes $f(\mathbf{x})$ in domain D. The set of function values is real and bounded from below.

The scope of application for simulation optimization is very vast. It covers various engineering tasks (e.g. optimal design of plants being designed), optimal control of existing systems (e.g. set point optimization) and combinatorial problems under uncertainty, to mention just a few. There exist many books (e.g. Rao, 1996) and articles (e.g. Carson and Maria, 1997; Stuckman *et al.*, 1991) on this subject. The aim of this paper is to communicate the latest developments in the field, and also to focus the attention at somewhat unpopular and difficult sort of problems: those with ill-posed constraints.

1.2 Problem characteristics

Using a simulator to compute an objective function value obviously does not, from strictly theoretical point of view, define by itself a special class of optimization problems. One can consider employing a simulator to compute the value of a linear function, in the extreme case. However, performing simulation has several practical reasons, preconditions and implications. Usually,

- The objective function is nonlinear (and, perhaps, discontinuous);
- The problem dimensionality is moderate ($\dim \mathbf{x} < 1000$);
- Most of the computation budget is consumed by the simulator, and very little by the optimization algorithm.

Therefore, only several types of algorithms are commonly used, those that are well suited to this particular kind of problems.

The major classification criteria for simulation optimization problems are the determinism or uncertainty of a model (problem) and continuity of the design variables. The next two features that affect the choice of an algorithm are problem constraints and the demand that the solution be globally optimal. Then, further classification of applied optimization algorithms may be done, depending on their ability to utilize extra information available from the simulation, and on the ability to balance between simulation time and accuracy. These topics will be shortly presented now.

1.2.1 Determinism and uncertainty

Determinism or uncertainty of a model is the main criterion for choosing the appropriate optimization algorithm. If $f(\cdot)$ in (1) depends only on the vector of design variables $\mathbf{x}$, then the problem is deterministic. If the simulation result is affected by a random vector, then the problem is stochastic. In the second case the objective function corresponding to $f(\cdot)$ in (1) is usually the expected value of $f(\cdot)$, and the optimization problem can be rewritten as follows:

$$\min_{\mathbf{x}\in D} \mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi}) \quad , \tag{2}$$

where $\boldsymbol{\xi}$ is some random variable.

1.2.2 Continuity of decision variables

A design variable may be allowed to change continuously or discretely (i.e. to take values from a set of discrete values), or both ways. If all decision variables are continuous, we have

a continuous optimization problem; if all decision variables are of the second kind, we have a discrete optimization problem. All remaining problems are classified as mixed. In reality, most of design problems are discrete due to standardization of available materials, discrete nature of actuators, and alike. Also measurements performed on a system are discrete for the same reasons. However, this fact by itself does not qualify all problems as discrete — such qualification is determined by the change in system behaviour caused by the smallest possible change of parameter value. If this change is not rapid (i.e. it is rather quantitative than qualitative) then continuous optimization algorithms may be applied, as the discretization of the algorithm output does not change the nature of a solution. Otherwise we really deal with a discrete problem. Discrete problems may be attacked using a suite of direct optimization algorithms mentioned later in this text. There exist also approaches, like branch-and-bound method, employing continuous algorithms for the discrete optimization — mainly for their effectiveness.

1.2.3 Constraints

Constraints are often welcome in optimization problems as they can significantly reduce the search space, thus accelerating the operation of an algorithm (cf. Papalambros, 1988, p. 387). A constraint may result from various reasons: range of control inputs of an object, safety of operation of a modelled system, model validity etc. They are inevitable in design problems; their lack usually means that some part of problem definition process has not been performed carefully enough.

All the constraints that are defined explicitly with respect to the decision variables, are desirable, since they can be either supported directly by an optimization routine (e.g. linear ones) or handled by appropriate transformations of the design variables (Papalambros, 1988, p. 383), or by penalty functions. Unfortunately, they also may be accompanied by a number of implicit constraints, i.e. those depending on the variables of the system that are modelled, which means that their values are known only after the simulation is performed successfully (if it finishes altogether). Those constraints, as more characteristic of a deterministic optimization, are discussed in Section 3.

1.2.4 Global optimality of solution

Practice shows that a person interested in optimal design done via simulation optimization is satisfied rather by a substantial improvement of the objective function than by its global optimality. It is so in the case when the initial solution already exists. Nevertheless, it is always welcome to have an optimization routine that looks for the global solution.

Globality of an optimization algorithm cannot be obtained without either rigorous assumptions about the objective function and the domain (both being convex, for example), or rather special arrangements concerning the algorithm and making it, usually, not very efficient. The latter case is particularly painful when the simulation times are long. Often the users, unable to make assumptions on $f(\cdot)$, finally resort to evolutionary strategies and other expensive routines (see e.g. Faccenda and Tenga, 1992).

1.2.5 Utilization of additional information

Every evaluation of $f(\cdot)$ requires running the simulator which is often an opaque piece of software (also called a black box). It means that there is no way of getting more information

about the simulation outcome than those made available by the software manufacturer. In practice, it means particularly that the derivative $\frac{\partial}{\partial \mathbf{x}} f(\cdot)$ is not directly available.

As the gradient information can speed up the optimization by an order of magnitude, there is always a strong need for it, especially in the stochastic optimization case. Few specialized simulators (called in Pflug, 1996, p. 19 white boxes) offer the feature of gradient computation, hence in most cases the optimizer has to estimate it by itself. Standard ways of gradient estimation are presented in Section 2.

1.2.6 Adaptation of simulation parameters

The internal working of most simulators is affected by model parameters. Those parameters influence such simulation features such as running time, overall accuracy, simulator internal rounding and discretization. The ability to accept different parameter values for each simulation run is the valuable feature of a simulator. The question is whether the optimizer can make the use of it. If so, rough simulation can be done in a preliminary phase of optimization, followed then by finer and still finer search as the solution is being approached. Formally, those general simulation parameters can be perceived as extra elements of the vector $\mathbf{x}$ of design variables.

2 Stochastic optimization

Let us assume now that the objective of stochastic optimization is to minimize the expected value of a given function of both design and random variables, i.e. as in (2). Such problem formulation dominates in the literature.

Virtually any deterministic method can be employed for stochastic optimization, provided that the function value and derivatives are replaced by their appropriate estimates (see Pflug, 1996, p. 22), but such an ample switch from the original algorithm to its stochastic counterpart is usually inefficient. Therefore, numerous algorithms have been designed specifically for stochastic optimization. For their overview, see e.g. (Pflug, 1996; Andradóttir, 1998b; Fu, 1994).

Optimization algorithms are classified in this paper according to the kind of information and assumptions they need to operate upon. An important category, called stochastic approximation methods, follows the well known Cauchy's steepest descent scheme that requires both objective function and gradient values to be available. The second branch are response surface methods which assume that $f(\cdot)$ belongs to a specific class of functions. Those algorithms which operate with only the objective value available, making no particular assumptions of $f(\cdot)$, form the third group of the direct search methods.

2.1 Stochastic approximation

Stochastic approximation methods solve continuous stochastic simulation optimization problems as formulated in (2) by utilizing a random estimate $\hat{\mathbf{g}}(\mathbf{x}, \boldsymbol{\xi})$ of the objective function gradient $\mathbf{g}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$. Algorithms of this type generate a sequence $\{\mathbf{x}_n\}$ of problem solution estimates, and the generic formula for one algorithm step is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - a_n \hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi}) \ , \tag{3}$$

where $\{a_n\}$ is a sequence of positive step sizes such that

$$\sum_{n=1}^{\infty} a_n = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} a_n^2 < \infty \ . \tag{4}$$

The above algorithm should converge to a point where $\hat{\mathbf{g}}(\mathbf{x}) = \mathbf{0}$, which does not necessarily have to be the minimizer of $\mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$. To ensure that the routine (3) indeed converges to the global minimum, additional assumptions are required, in particular that the objective function and the optimization domain are convex. This, however, does not yet ensure the proper convergence if the objective grows faster than quadratically when $\mathbf{x}$ changes. Moreover, this basic optimization scheme still does not support the constraints on $\mathbf{x}$. Also, the proper choice of $\{a_n\}$ affects the algorithm convergence.

2.1.1 Gradient estimation

There exist three basic gradient estimation procedures: finite differences, infinitesimal perturbation analysis (IPA) and likelihood ratio method (LR). For another one, the frequency domain experimentation, see (Jacobson, 1994).

Applying the method of finite differences, as requiring the least information of $f(\cdot)$, is often the only possible approach. However, this estimator has a big variance, is biased and computationally demanding. The estimate is obtained either by forward difference

$$\hat{\mathbf{g}}(\mathbf{x}, \boldsymbol{\xi}) = \sum_{i=1}^{\dim \mathbf{x}} \mathbf{e}_i \frac{f(\mathbf{x} + c\mathbf{e}_i, \boldsymbol{\xi}) - f(\mathbf{x}, \boldsymbol{\xi})}{c} \tag{5}$$

or by central difference formula

$$\hat{\mathbf{g}}(\mathbf{x}, \boldsymbol{\xi}) = \sum_{i=1}^{\dim \mathbf{x}} \mathbf{e}_i \frac{f(\mathbf{x} + c\mathbf{e}_i, \boldsymbol{\xi}) - f(\mathbf{x} - c\mathbf{e}_i, \boldsymbol{\xi})}{2c} \ . \tag{6}$$

Here, $\mathbf{e}_i$ denotes a versor along the i-th axis. Formulas (5) and (6) require $\dim \mathbf{x} + 1$ and $2 \dim \mathbf{x}$ evaluations of $f(\cdot)$, respectively. The optimization scheme (3) with a finite differences formula applied for gradient estimation is known as Kiefer-Wolfowitz (KW) algorithm.

An important improvement to the above scheme was proposed in (Spall, 1992). Instead of $\dim \mathbf{x} + 1$ or $2 \dim \mathbf{x}$ evaluations of $f(\cdot)$, it requires only $2p$ evaluations, p being considerably smaller than $\dim \mathbf{x}$. The idea is not to make an estimation of $\hat{\mathbf{g}}(\cdot)$ in each direction by separate shifts of $\mathbf{x}$, but to perturb $\mathbf{x}$ simultaneously in all directions p times, and then to calculate the mean

$$\hat{\mathbf{g}}(\mathbf{x}, \boldsymbol{\xi}) = \frac{1}{-} \sum_{i=1} \left(\frac{f(\mathbf{x} + c\boldsymbol{\Delta}_i, \boldsymbol{\xi}) - f(\mathbf{x} - c\boldsymbol{\Delta}_i, \boldsymbol{\xi})}{2c} \sum_{j=1}^{\dim \mathbf{x}} \frac{\mathbf{e}_j}{\langle \mathbf{e}_j, \boldsymbol{\Delta}_i \rangle} \right) \ . \tag{7}$$

In (7), $\boldsymbol{\Delta}_i$ is a realization of some zero-mean random variable, and c is a scaling factor (actually, c is not constant but predetermined for each algorithm step n, usually $c_n = c/n^{\gamma}$, $c, \gamma > 0$). By applying simultaneous perturbation gradient approximation method, significant speed-up of computations can be achieved, especially for moderate and large scale problems. For an exemplary practical application of this routine, see (Kleinman *et al.*, 1998).

p

IPA and LR methods do not require running dedicated simulations to estimate the performance gradient. Both procedures are based on the observation that the present results of, say N, simulations (i.e. N realizations of $\boldsymbol{\xi}$) can be used not only for computation of the performance value but for computation of the gradient as well. Calculation of the performance value usually involves some kind of averaging, and therefore the intermediate results (e.g. "paths" of a discrete event dynamic system) are left unused by the finite differences method. In IPA and LR, by the careful inspection of those results one can infer the behaviour of the system if $\mathbf{x}$ were going to change. The advantage of IPA and LR is that they often yield unbiased and consistent estimates, but they require knowledge of the structure of the simulated stochastic system — namely the cumulative distribution function for the random variable $\boldsymbol{\xi}$ is supposed to be known.

Formally, both IPA and LR start off conceptually from the formula for the gradient of the performance function:

$$\mathbf{g}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi}) = \frac{\partial}{\partial \mathbf{x}} \int f(\mathbf{x}, \boldsymbol{\xi}) p(\mathbf{x}, \boldsymbol{\xi}) d\boldsymbol{\xi} = \int p(\mathbf{x}, \boldsymbol{\xi}) \frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}, \boldsymbol{\xi}) d\boldsymbol{\xi} + \int f(\mathbf{x}, \boldsymbol{\xi}) \frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}, \boldsymbol{\xi}) d\boldsymbol{\xi} \quad , \tag{8}$$

where $p(\cdot)$ is probability density function for the random variable $\boldsymbol{\xi}$. Since in general both $f(\cdot)$ and $p(\cdot)$ can depend on $\mathbf{x}$, the formula splits finally into the sum of two integrals. The former integral can be estimated by computing the mean $\frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}, \boldsymbol{\xi}_i)$ for the same realizations of $\boldsymbol{\xi}$ that were used in performance computation. The handling of the latter depends on the method: IPA transforms the problem to make it zero altogether, while LR computes it utilizing still the same realizations of $\boldsymbol{\xi}$. In LR (also known as score function method), the second integral is rewritten as follows:

$$\int f(\mathbf{x}, \boldsymbol{\xi}) \frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}, \boldsymbol{\xi}) d\boldsymbol{\xi} = \int f(\mathbf{x}, \boldsymbol{\xi}) \frac{p(\mathbf{x}, \boldsymbol{\xi})}{p(\mathbf{x}, \boldsymbol{\xi})} \frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}, \boldsymbol{\xi}) d\boldsymbol{\xi} = \int p(\mathbf{x}, \boldsymbol{\xi}) \left(f(\mathbf{x}, \boldsymbol{\xi}) \frac{\frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}, \boldsymbol{\xi})}{p(\mathbf{x}, \boldsymbol{\xi})} \right) d\boldsymbol{\xi} \tag{9}$$

so as to resemble the first integral in (8). Now it can be estimated by computing the mean $\frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}, \boldsymbol{\xi}_i) \frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}, \boldsymbol{\xi}_i) / p(\mathbf{x}, \boldsymbol{\xi}_i)$ for the same realizations of $\boldsymbol{\xi}$ as previously.

IPA represents $f(\mathbf{x}, \boldsymbol{\xi})$ with $\boldsymbol{\xi}$ of any type (in particular, depending on $\mathbf{x}$) in an alternative way, by a function $F^{-1}(\mathbf{x}, \mathbf{u})$ such that $\mathbf{u}$ is a random variable uniformly distributed in $[0, 1]^{\dim \mathbf{u}}$ (and, certainly, independent of $\mathbf{x}$). Under these conditions, and following the rules from (8), we have:

$$\mathbf{g}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \mathbf{E}_{\boldsymbol{\xi}} F^{-1}(\mathbf{x}, \mathbf{u}) = \frac{\partial}{\partial \mathbf{x}} \int F^{-1}(\mathbf{x}, \mathbf{u}) d\mathbf{u} = \int \frac{\partial}{\partial \mathbf{x}} F^{-1}(\mathbf{x}, \mathbf{u}) d\mathbf{u} \quad . \tag{10}$$

The above equation is simpler than formulas for LR because all the analytical effort is hidden in the formulation of F^{-1}, which is possible only if one knows how to make $\boldsymbol{\xi}$ out of $\mathbf{u}$. Next, differentiating F^{-1} resembles tracing system behaviour in presence of infinitesimally small changes of $\mathbf{x}$, and that is why IPA is called so. The gradient in IPA is approximated by computing the mean $\frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{x}} F^{-1}(\mathbf{x}, \mathbf{u}_i)$, where $\mathbf{u}_i$ are realizations of $\mathbf{u}$, analogously to $\boldsymbol{\xi}$.

LR is applicable to a larger class of problems than IPA, but also the estimates obtained from it tend to have larger variances. The optimization scheme (3) with any unbiased gradient estimate applied is known as Robbins-Monro (RM) algorithm. For a comprehensive overview of gradient estimation methods, see e.g. (Andradóttir, 1998a, p. 312) or (Pflug, 1996, p. 231).

2.1.2 Variance reduction

Certainly, keeping the variance of both objective and gradient estimate as small as possible improves the algorithm convergence. To achieve this, the following variance reduction techniques are used: control variates, conditioning, stratified sampling, importance sampling and antithetic random variables. They all are based on the same idea of utilizing some information about the model to reduce the variance of the estimator. Different techniques use different information to reduce the variance. Control variates technique redefines the estimator of $\mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$, involving in it some other random variable $\mathbf{z}$, observable during simulation, whose expected value is known exactly. By appropriate coefficient setting one can get reduction of the variance, provided that $f(\cdot)$ and $\mathbf{z}$ are related. Conditioning technique is applied when the process of simulating the value of $f(\cdot)$ can be split logically in two stages: first, when some random variable $\mathbf{y}$ can be observed, and second, when $f(\cdot)$ is generated from $\mathbf{y}$ using some known conditional distribution. Instead of observing only the final result, one observes $\mathbf{y}$ and directly calculates the conditional expectation. Stratified sampling technique is in some way complementary to conditioning: here the detailed distribution of the intermediate observation $\mathbf{y}$ is known, and the simulation needs to be started only to complete the calculation of $f(\cdot)$. In practice the completing simulation is run once for each of the values that $\mathbf{y}$ can take and its results, along with the known distribution of $\mathbf{y}$, serve to calculate the estimator of $\mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$. Importance sampling technique is used in cases when events that are unlikely in the simulation process contribute significantly the value of $\mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$, e.g. in Monte Carlo integration of peaky functions. A new sampling density is chosen that puts more weight to an area that affects the estimator of $\mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$ strongly. Antithetic random variables technique uses, in subsequent simulation runs, random variables having the same distributions as $\boldsymbol{\xi}$, but being correlated in such way that the estimator variance be reduced. For example, let $f(\cdot)$ be a monotonic function of $\boldsymbol{\xi} = (\xi_1, \xi_2, \dots)$, $\xi_i \sim \mathcal{U}(0,1)$ — then having a single realization of $\boldsymbol{\xi}$ one may run two simulations: one with $(\xi_1, \xi_2, \dots)$ and another with $(1-\xi_1, 1-\xi_2, \dots)$ that are negatively correlated, and hope that the same is true about $f(\cdot)$. For broader description of variance reduction techniques, see e.g. (Pflug, 1996, p. 221).

Except from the above general techniques of variance reduction, for the purpose of comparing different systems there is a widely used practice of utilizing the same realization of random variable $\boldsymbol{\xi}$ in all steps of algorithm (3), known as common random numbers scheme, and also referred to as sample path optimization. Having applied the common random numbers, the optimization problem becomes, in fact, a deterministic one, and can be solved by a variety of methods. Many methods want the number of elements of $\boldsymbol{\xi}$ to grow as the current solution nears the optimum $\mathbf{x}^\star$ to increase the simulation accuracy. The proof that (under certain assumptions) the solution $\mathbf{x}^{\star,\dim\boldsymbol{\xi}}$ of the corresponding deterministic problem (based on a sample path of length $\dim\boldsymbol{\xi}$) tends to $\mathbf{x}^\star$ for growing $\dim\boldsymbol{\xi}$, is given in (Robinson, 1996). For most of simulation optimization problems, various size of $\boldsymbol{\xi}$ can be handled, and has a reasonable interpretation. If, for example, $f(\cdot)$ is the output from Monte Carlo integration routine, then $\dim\boldsymbol{\xi}$ is the number of sample points used for the integration. In another example, where $f(\cdot)$ is the performance of a queuing system, $\dim\boldsymbol{\xi}$ is the number of interarrival times, i.e. the number of served customers.

Robinson (Robinson, 1996) discusses general convergence properties of sample path algorithms with constant $\dim\boldsymbol{\xi}$; instead Shapiro and Wardi (Shapiro and Wardi, 1996a) consider the plain stochastic approximation scheme, but with a vector of common random numbers

being periodically augmented with new random elements. This operation improves accuracy of estimates as the algorithm is getting closer to the solution $\mathbf{x}^\star$, but restores randomness to the problem. See also (Shapiro and Wardi, 1996b) for a study on convergence of methods working in this fashion, especially in the case when $f(\cdot)$ cannot be estimated otherwise than by Monte Carlo simulation.

2.1.3 Adaptation of the step size and precision

The question arises how to choose the precision of simulation (e.g. number N of points for Monte Carlo simulation) and the sequence $\{a_n\}$ of step sizes to maintain reasonable compromise between the solution accuracy and the computing effort. The initial technique is to use a fixed simulation precision, and to define $\{a_n\}$ *a priori*:

$$a_n = \frac{a}{n} \ , \tag{11}$$

where a is some initial step size. It is useful to consider another formula:

$$a_n = \frac{b}{c+n} \quad \text{where} \quad c \gg b > 0 \ , \tag{12}$$

which prevents a_n from decreasing rapidly for small n. For (11), it is shown in (Pflug, 1996, p. 290) that if we consider more general formula, $a_n = a/n^\alpha$, then $\alpha = 1$ will still be the best choice.

The same author proposes an adaptive procedure for choosing a_n, based on the observation that the step size is optimal if the gradients in two consecutive steps stay orthogonal, i.e. $\langle \hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi}), \hat{\mathbf{g}}(\mathbf{x}_{n+1}, \boldsymbol{\xi}) \rangle = 0$. The proposed routine reduces a_n by half if the above dot product is going to be negative. Of course, for the above formula to make sense, the gradient estimates have to be based on the same realization of $\boldsymbol{\xi}$, at least for the purpose of comparing $\hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi})$ and $\hat{\mathbf{g}}(\mathbf{x}_{n+1}, \boldsymbol{\xi})$.

The postulate for adaptation of a_n can be found in many papers (see e.g. Wardi, 1990; Shapiro and Wardi, 1996b; Yan and Mukai, 1993). The general suggestion for a_n to be the result of exact line minimization along $\hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi})$ in practical implementations usually is replaced by the more liberal Armijo rule that in the case of algorithm (3) is:

$$|\langle \hat{\mathbf{g}}(\mathbf{x}_n - a_n \hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi}), \boldsymbol{\xi}), -\hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi}) \rangle| \leq -\epsilon \, \langle \hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi}), -\hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi}) \rangle \ . \tag{13}$$

This means that the product of the minimization direction $-\hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi})$ and the gradient estimate at $\mathbf{x}_{n+1}$ does not have to be zero (as in exact minimization case) but a small (delimited by ϵ) part of an analogous product at $\mathbf{x}_n$.

The authors cited above combine adaptation of a step size with the increasing simulation precision. Wardi and Shapiro (Wardi, 1990; Shapiro and Wardi, 1996b) assume that the size of the random variable (i.e. the number of random points passed to the simulator) grows to the infinity in a predetermined way as the algorithm proceeds, and show the convergence in this general case. Next, Yan and Mukai (Yan and Mukai, 1993) propose the measures to monitor the progress of performance optimization and of the estimation error, and use them to determine when the size of random variable $\boldsymbol{\xi}$ should be increased. The progress measure is based on the absolute difference $|f(\mathbf{x}_n, \boldsymbol{\xi}) - f(\mathbf{x}_{n+1}, \boldsymbol{\xi})|$ of the objective at two consecutive solution estimates for the same realization of the random variable, and the error measure is

based on the absolute difference $|f(\mathbf{x}_n, \boldsymbol{\xi}) - f(\mathbf{x}_n, [\boldsymbol{\xi}, \boldsymbol{\xi}^+])|$ of objective at the same solution estimate, but for a realization of random variable $\boldsymbol{\xi}$ alone, and for $\boldsymbol{\xi}$ augmented with new random elements $\boldsymbol{\xi}^+$.

The simulation accuracy, and the underlying size of random variable used to estimate the objective and its gradient, depend on the kind of simulation. In this section mostly Monte Carlo simulations were considered. However, (Chong and Ramadge, 1993) discusses the case of optimizing the service time in a single server queue with generally independent customer interarrival time (GI/G/1). It is proved that the next solution estimate $\mathbf{x}_{n+1}$ can be computed as soon as a new customer arrives, and the algorithm still converges.

The ideas of step size and simulation accuracy adaptation are particularly important when an approximation of the objective function is made which is more complex than linear. Methods based on such approximations are discussed in Sec. 2.1.6.

2.1.4 Averaging

A very simple yet powerful modification to the classic stochastic approximation algorithm that dramatically improves its convergence, was simultaneously invented by Polyak and Ruppert. It is supported by appropriate proofs in (Polyak and Juditsky, 1991), and one of numerical examples for its superiority can be found in (Yin, 1991). The idea is that instead of observing a sequence $\{\mathbf{x}_n\}$ of solution estimates, one can observe a sequence $\{\overline{\mathbf{x}}_n\}$ of their averages

$$\overline{\mathbf{x}}_n = \frac{1}{n} \sum_{i=0}^{n} \mathbf{x}_i \quad . \tag{14}$$

In this case, however, to preserve the algorithm's convergence, one has to replace the formula (11), or (12), with

$$a_n = \frac{a}{n^\alpha}, \quad 0 < \alpha < 1 \quad . \tag{15}$$

The idea of averaging has been developed further in (Kushner and Yang, 1993). The authors propose that the averaging does not have to be performed over the whole history. They prefer to calculate a moving average

$$\overline{\mathbf{x}}_n = \frac{1}{w_n + 1} \sum_{i=n-w_n}^{n} \mathbf{x}_i \tag{16}$$

and to set the averaging window size w_n so that the convergence is still preserved. Generally, w_n depends on α in (15), i.e. w_n decreases as $\alpha \to 0$ because bigger oscillations of $\{\mathbf{x}_n\}$ around $\mathbf{x}^\star$ need less averaging.

Yet another interesting adaptation of an idea of averaging is presented in (Delyon and Juditsky, 1993). It is proposed to apply the basic averaging scheme (14) with the sequence of step sizes defined *a priori* but indexed in other way than just with an index of the current step. The indexing variable there is the counter of "flips" of $\hat{\mathbf{g}}(\cdot)$ that happen nearby $\mathbf{x}^\star$. Therefore, the algorithm reduces its step size only in the final stage, while a vicinity of the solution has been reached.

2.1.5 Constraints

Stochastic approximation algorithms are definitely not robust. Strong assumptions are made in the literature with respect to both the objective function $f(\cdot)$ and the optimization

domain D. To ensure global convergence, $f(\cdot)$ is required to be strictly convex, and D is required to be closed and convex. Usually, it is assumed that D is a cartesian product of the intervals of feasible values for each x_i. The common approach to handle constraints is to project the current estimate of $\mathbf{x}^\star$ on D in each step of the routine. Therefore, (3) would be replaced by

$$\mathbf{x}_{n+1} = \pi_{n+1}\left(\mathbf{x}_n - a_n \hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi})\right) \quad , \tag{17}$$

where π_n is a constraint preserving operator.

Usually, $\pi_{n+1}(\mathbf{x})$ accepts $\mathbf{x}$ if $\mathbf{x} \in D$, and preserves $\mathbf{x}_n$ otherwise (for an example of such algorithm see (Chong and Ramadge, 1993)). For another type of projection, recall (Wardi, 1990) — there, the domain D is defined through a set of functions $h_i(\cdot)$, $h_i(\mathbf{x}) \leq 0$, and a projection is integrated into the line minimization procedure: the step size a_n must be chosen so that none of $h_i(\mathbf{x}_n - a_n \hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi}))$ becomes positive.

There is another, mentioned earlier, reason, for which the basic stochastic approximation scheme is so widely supplemented with the projection feature. It is because (3) does not converge if $f(\cdot)$ grows superlinearly. For badly chosen a and the starting point $\mathbf{x}_0$, the rate at which $\{a_n\}$ decreases is simply insufficient to overcome the growth of $\mathbf{g}(\cdot)$ and the values of $\{\mathbf{x}_n\}$ start to oscillate around the solution. It is not so rare a case: Fu (Fu, 1990) considers one-dimensional parameter optimization problem for a GI/G/1 queue with the projection operator

$$\pi(x) = \begin{cases} x_{min} & \text{if } x < x_{min} \\ x_{max} & \text{if } x > x_{max} \\ x & \text{otherwise} \end{cases} \quad , \tag{18}$$

where x_{min} and x_{max} are the bounds. Unfortunately, the projection operator (18) does not suffice because in the considered case $f(x) \to \infty$ as $x \to x_{min}$ or $x \to x_{max}$. It is proposed to solve this difficulty by optimizing over a region that is smaller than $[x_{min}, x_{max}]$. Such workaround causes another problem: how to reduce D so that $x^\star$ still remain in the search domain?

A solution to this problem would be to change the size of the region on which $\mathbf{x}$ is projected in adaptive way. The proposal of such an adaptive projection was presented in (Chen and Zhu, 1986). First, a sequence $\{M_n\}$ of increasing radii is constructed, and a counter σ that stores the number of steps in which the values $\mathbf{x}_n - a_n \hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi})$ fall off the circle $C(\mathbf{0}, M_\sigma)$. In a case when the violation of this circular constraint would happen, $\mathbf{x}_{n+1}$ is reset to some predefined point $\mathbf{x}^+$. It means that the algorithm starts over from $\mathbf{x}^+$, but with a_n smaller and M_σ bigger than before, as the values of n and σ persist over algorithm restarts. By choosing $\{a_n\}$ and $\{M_n\}$ skillfully good convergence can be obtained. The idea presented in (Chen and Zhu, 1986) was further developed in (Andradóttir, 1995): the author improves the algorithm working in the case when $\mathbf{x}_{n+1}$ is reset to $\mathbf{x}^+$. He proposes, instead of returning to $\mathbf{x}^+$, to project $\mathbf{x}_n - a_n \hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi})$ onto a predefined set. Proceeding in this way, not all progress made in the previous iterations is lost.

2.1.6 Other improvements

Since 1951, the classic stochastic approximation algorithm has been modified and improved by many authors in other ways than discussed in the paragraphs above. Perhaps the most often adopted modification was to assume the step size a_n not to be a scalar but rather

a matrix, like in the well known Newton's method:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}^{-1}(\mathbf{x}_n)\frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}_n) \ , \tag{19}$$

where $\mathbf{J}(\mathbf{x}_n)$ is the Hessian matrix of $f(\cdot)$ at $\mathbf{x}_n$. Such improvement was proposed in (Ruppert, 1985; Wei, 1987), and claimed to accelerate algorithm convergence. Actually, the authors have assumed that $\frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}_n)$ was observable directly, and formulated the optimization problem as finding $\mathbf{x}^\star$ such that $\frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}^\star) = \mathbf{0}$. Their formula for $\{\mathbf{x}_n\}$ merges (3), (11) and (19) as follows:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \frac{a}{n}\hat{\mathbf{D}}_n(\mathbf{x}_n, \boldsymbol{\xi})\hat{\mathbf{g}}(\mathbf{x}_n, \boldsymbol{\xi}) \ , \tag{20}$$

where $\hat{\mathbf{D}}_n(\mathbf{x}_n, \boldsymbol{\xi})$ is an estimate of $\mathbf{J}^{-1}(\mathbf{x}_n)$ at $\mathbf{x}_n$ computed using finite differences scheme.

The latter paper proposes two more additions to (20). First, the estimation of $\mathbf{D}_n$ is based also on the values of $\mathbf{D}$ in the previous steps. Second, the identity matrix is used in steps where the Hessian matrix would be singular.

The paper (Plambeck *et al.*, 1996), mentioned already, proposes an entirely different algorithm in which the quadratic problem (QP) is solved in each step to find $\mathbf{x}_{n+1}$. The quadratic problem is deprived of any randomness, because all the necessary computations in a single step are performed for the same realisation of $\boldsymbol{\xi}$. The proposed approach is to find the subgradients $\mathbf{s}_1$, $\mathbf{s}_2$,... of $f(\cdot)$ at $\mathbf{x}_n$, thus constructing a cutting-plane approximation of the performance function. The problem passed to a QP solver is to minimize this approximation, which is additionally regularised so that $\mathbf{x}_{n+1}$ be not too far from $\mathbf{x}_n$. Therefore the routine is as follows:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{d} \qquad \mathbf{d}, v = \arg\min \left[v + \frac{||\mathbf{d}_n||^2}{t_n}, \ \ v >= \mathbf{s}_i^T \mathbf{d} \right] \ , \tag{21}$$

where t_n is the regularisation coefficient. QP also allows to support any other equality or inequality constraints that have been imposed on the original problem externally. Application of subdifferential broadens the suite of problems supportable by the algorithm.

A significant improvement of optimization efficiency can be achieved not only through the algorithm amendments, but also by simplifying the simulator itself. Let us return once again to (Plambeck *et al.*, 1996) for an example. The authors consider discrete tandem production lines, i.e. those producing separate workpieces. However, they use simulation routines suitable for the continuous products like fluids. By doing so, they reduce the simulation time by an order of magnitude at the cost of 4% error in $f(\cdot)$ relative to the discrete simulation.

While certain researchers' goal is to reduce the variance, others deliberately make the optimization routines wander for some time in D in search for the global, not local, solution. Such an algorithm is presented in (Gelfand and Mitter, 1991); there every new solution estimate is perturbed with a multidimensional independent Gaussian random variable. The variance of this variable diminishes as the algorithm proceeds, making it less and less able to leave the attraction area of a local minima. This algorithm resembles the simulated annealing scheme, described later.

2.2 Response surface methods

Unlike in stochastic approximation, in the classic response surface methods there is no interaction between the optimizer and the simulator. The optimization is performed in three phases:

1. Simulation is invoked at a certain number of points from the domain D. Those points can be chosen at random or using some predefined scheme. The results of simulations are passed to Phase 2.

2. The unknown function $\mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$ is approximated by some known deterministic function $\tilde{f}(\mathbf{x})$ by regression, using the values calculated in Phase 1.

3. The minimizer of $\tilde{f}(\mathbf{x})$ is sought by some optimization routine.

In most practical applications, phases 1–3 are executed more than once. For subsequent executions, as the solution estimate approaches $\mathbf{x}^\star$, the functions $\tilde{f}(\cdot)$ of more complex structure are being fitted (e.g. first linear, then quadratic, finally cubic). Also the region from which the samples of $f(\cdot)$ are drawn, becomes tight. According to (Carson and Maria, 1997), the response surface methods perform in general better than their stochastic approximation alternatives.

2.3 Direct search methods

The algorithms discussed in this section are directly value based, i.e. their behaviour is based only on the value of the objective, without any support of the objective derivatives. On one hand, the absence of such support must definitely have an adverse impact on algorithm effectiveness, especially for medium and large scale problems. On the other, it improves the algorithm robustness.

Fortunately, in many practical problems the dimensionality is moderate or it can be made moderate by some sort of aggregation. Then one can think of applying the simplest imaginable direct search method: the random search technique. This technique works by choosing at random the points from D, keeping the record of best the one found so far. However, due to the randomness of $f(\cdot)$, the best point found by this algorithm may still happen to lie quite far from $\mathbf{x}^\star$. Like in stochastic approximation methods, some kind of averaging must be employed to mitigate the fluctuations of $f(\cdot)$.

2.3.1 Simplex search

The averaging may be accomplished not only by repeating simulation several times for the same $\mathbf{x}$. Another approach consists in maintaining a pool $X_n = \{\mathbf{x}_{n,1}, ..., \mathbf{x}_{n,m}\}$ of m trial designs in each algorithm step so that simulation is performed once for each $\mathbf{x}_{i,j}$. Then, the pool in the next step is based on the current pool contents.

Such algorithm was proposed in (Spendley *et al.*, 1962). It is a classic simplex search routine with adaptations for stochastic optimization problem. The algorithm in consecutive steps moves, by reflections, a simplex of trial points until proximity of $\mathbf{x}^\star$ is reached. Then the simplex starts to wander around the solution which is the sign to stop the procedure. Without any improvements that original simplex algorithm would probably get anchored

far from $\mathbf{x}^\star$ by an occasional unusually good simulation result. To avoid that, the authors propose to repeat simulation only for "stale" simplex vertices to force the algorithm forward.

The simplex search, endowed with expansion/contraction operations and known as the Nelder-Mead procedure, is a widely applied routine. However, when employed for stochastic problems, it terminates prematurely due to far too frequent contractions it happens to make. This drawback is addressed in (Barton and Ivey, Jr., 1991). The authors, after a brief review of the former algorithm amendments, point out the cases where resampling of $f(\cdot)$ solves the problem. They additionally advice making the contraction operation less strong. Yet another revised simplex procedure is proposed in (Humphrey and Wilson, 1998), and numerical examples are provided showing its superior performance relative to the algorithm presented in (Barton and Ivey, Jr., 1991). The postulated changes include geometrical decrease (resp. linear growth) of the contraction (resp. expansion) coefficients and procedure restarting (with preserving the actual best solution) as the routine proceeds.

Simplex search routines, like all routines where the selection of a trial point is driven by ranking of points in the pool, work well if the fluctuations of $f(\cdot)$ due to random factors are so small that they do not affect the ranking of the pool points. It is usually the case in the preliminary stage of optimization. When approaching $\mathbf{x}^\star$, the ordering of points in X_n becomes distorted by the randomness of $\boldsymbol{\xi}$, and the algorithms start oscillating around the solution.

2.3.2 Simulated annealing

Unlike simplex search, the simulated annealing algorithm works with just one point only at each step which is the current solution approximation $\mathbf{x}_n$. It makes attempts to improve it by choosing at random some candidate point $\tilde{\mathbf{x}}_{n+1}$ in the neighbourhood of $\mathbf{x}_n$, and eventually accepting it according to the following formula

$$\mathbf{x}_{n+1} = \begin{cases} \tilde{\mathbf{x}}_{n+1} & \text{if } f(\tilde{\mathbf{x}}_{n+1}) < f(\mathbf{x}_n), \\ \tilde{\mathbf{x}}_{n+1} & \text{with probability } e^{-\frac{f(\tilde{\mathbf{x}}_{n+1})-f(\mathbf{x}_n)}{T_n}} \text{ if } f(\tilde{\mathbf{x}}_{n+1}) > f(\mathbf{x}_n), \\ \mathbf{x}_n & \text{with probability } 1 - e^{-\frac{f(\tilde{\mathbf{x}}_{n+1})-f(\mathbf{x}_n)}{T_n}} \text{ if } f(\tilde{\mathbf{x}}_{n+1}) > f(\mathbf{x}_n), \end{cases} , \tag{22}$$

where T_n is the current "temperature", i.e. a coefficient allowing the algorithm to climb uphill in search for global optimum. T_n must decrease to zero as the algorithm proceeds.

Adaptations of algorithm (22) to stochastic problems use an estimate of the objective function value (mostly it is the mean of an increasing number of observations of $f(\cdot)$ at $\tilde{\mathbf{x}}$). They are also *greedy*, i.e. they preserve the best solution estimate found in the course of the algorithm run. Convergence properties of such algorithm are discussed by Gelfand and Mitter (Gelfand and Mitter, 1989). Next, Fox and Heine (Fox and Heine, 1995) postulate some amendments that prevent simulated annealing from staying too long without a move, thus improving its efficiency.

An adaptation of the number of observations used to estimate the objective is proposed by Bulgak and Sanders (Bulgak and Sanders, 1988). They postulate that the number of observations must be increased only if the difference $f(\tilde{\mathbf{x}}_{n+1}, \boldsymbol{\xi}) - f(\mathbf{x}_n, \boldsymbol{\xi})$ is small w.r.t the current confidence intervals for $f(\tilde{\mathbf{x}}_{n+1}, \boldsymbol{\xi})$ and $f(\mathbf{x}_n, \boldsymbol{\xi})$. Moreover, if the replication of observations of $f(\cdot)$ does not allow for reliable comparison, then the temperature is reduced and another trial point $\tilde{\mathbf{x}}_{n+1}$ is chosen. Such algorithm has been successfully applied for an exemplary problem of buffer sizes optimization in transport systems. A similar algorithm is

also used in (Barretto *et al.*, 1999), and applied for steelworks design parameters (number of cranes, furnaces etc.) problem. In both cases the search domain was discrete, and a proper neighbourhood had to be constructed so that the algorithm could penetrate the whole search space.

Generally speaking, simulated annealing can be used in stochastic optimization, both continuous and discrete, provided that the variance of the estimator of the objective function decreases to zero for growing number of observations. Also, proper definition of the neighbourhood is an important issue.

2.3.3 Other algorithms

An algorithm quite similar to simulated annealing, and frequently cited, has been presented in (Yan and Mukai, 1992). Instead of dealing with the randomness of the objective function by averaging of an increasing number of samples, the authors propose to utilize the random character of the objective for the purpose of searching for the global optimum. In fact, the needed modifications of the basic simulated annealing scheme are few: the main is that instead of looking for the minimum of $\mathbf{E}_{\boldsymbol{\xi}} f(\mathbf{x}, \boldsymbol{\xi})$ one looks for such $\mathbf{x}$ that maximizes the probability that $f(\mathbf{x}, \boldsymbol{\xi}) \leq \vartheta$. Here, ϑ is a random variable uniformly distributed over the interval $[f_{min}, f_{max}]$ of approximate values that $f(\cdot)$ can take. In practical realization of this algorithm the value $f(\tilde{\mathbf{x}}_{n+1}, \boldsymbol{\xi})$ is measured M_n times against a sort of a "stochastic ruler", i.e. against ϑ. If in all measurements the test $f(\tilde{\mathbf{x}}_{n+1}, \boldsymbol{\xi}) \leq \vartheta$ is passed, then the candidate $\tilde{\mathbf{x}}_{n+1}$ is accepted. Of course, for M_n growing with the progress of optimization, the tendency to accept worse candidate points decreases, analogously to the classic simulated annealing scheme. The proposed algorithm converges under relatively mild conditions, and is simple to implement.

Another, still more general and more global algorithm was proposed in (Andradóttir, 1996). This routine is also based on the simulated annealing scheme (or rather, on the random walk scheme). It operates on the domain D being a set of a finite number of points and for all the elements of D the probability of being chosen as the next trial point $\tilde{\mathbf{x}}_{n+1}$ is the same. Therefore, no cooling scheme does exist, but for each point in D the number of times that this point has been visited, is recorded. As the algorithm proceeds, the statistics are collected, and the point visited most often is considered to be the solution. In this routine the candidate $\tilde{\mathbf{x}}_{n+1}$ is accepted to be $\mathbf{x}_{n+1}$ only if $f(\tilde{\mathbf{x}}_{n+1}, \boldsymbol{\xi}) < f(\mathbf{x}_n, \boldsymbol{\xi})$. The convergence of this algorithm to the global optimum is demonstrated.

Quite often the engineers and researchers, inspired by widely recognized classic algorithms, develop their own routines, often tailored to the specifics of a particular problem. Those routines can combine the techniques existent since long time in separate algorithms, into one piece of software. An example of such approach can be found in (Svensson, 1997), where the problem of optimal network design is addressed. The author employs common random numbers scheme (to reduce the estimator variance), adaptation of sample size (to increase accuracy), perturbation analysis and, finally, the tabu search paradigm, obtaining a method that performs very well, at least for the considered class of problems.

There are many other well known and widely applied routines for stochastic optimization problems: evolutionary strategies, tabu search, importance sampling methods. For a short discussion on them and for further references, see (Andradóttir, 1998b; Carson and Maria, 1997; Stuckman *et al.*, 1991). Some of them will also be mentioned in Section 3, as they are suitable to cope not only with randomness, but also with other difficulties caused by the

simulators.

3 Deterministic optimization

This section addresses difficulties other than randomness that emerge in simulation-based optimization problems. It does not mean that in the cases considered below simulation always yields a result not distorted by some unpredictable value — in reality this happens rarely. Yet, the assumption can be made that the influence of uncertainty on the workings of the algorithm is negligible. Here, constraints and the shape of the objective function surface are of the most concern.

3.1 Implicit constraints

Apart from being specified explicitly by an open formula, the search domain D may also be defined not directly. Consider a case where, as a result of simulation, the value of a vector of dependent variables $\mathbf{v} = (v_1, v_2, \dots)$ is computed. Let the elements of $\mathbf{v}$ be subject to the box constraints, $v_{i,min} \leq v_i \leq v_{i,max}$ (one can also consider some more general open-form constraint specification for $\mathbf{v}$). Generally, there is no way of mapping those constraints onto D. This problem appears, for example, in control optimization for multilevel systems, and was reported in (Findeisen *et al.*, 1980, p. 36). In this case solving local optimization problems in the lower layer is the analogue of the simulation — the local solutions may turn out to be infeasible, and, usually, no general prescription exists for how to choose coordination decisions so that the box constraints on $\mathbf{v}$ are satisfied.

Determining D may become even more troublesome if the simulation does not yield any measure of by how much the constraints on $\mathbf{v}$ are violated. It is either because the simulation fails altogether (see Kamola and Malinowski, 2001, for an example) or it gives some output which is then disqualified by a verification procedure. In such case no indications as to the direction towards the feasible region can be derived from the simulator.

3.2 Shape of the response surface

Steep slopes, plateaux burrowed with narrow valleys, saddle points, discontinuities — simulation-based objective functions raise difficulties of all sorts for optimization routines. These unpleasant features appear partially due to imprecise model understanding, resulting in internal switching inside the simulator, partially due to finite computation precision and rounding, and partially because such is the real nature of the simulated system (recall Fu, 1990). For an example of such an objective function, refer to (Kamola and Miazga, 2001) where a problem of a microwave guide design is described.

3.3 Present approaches

In the literature can be found many guidelines for optimization problem preparation. As the constraints are regarded, most authors advise to take a closer look at the model first (see Papalambros, 1988, pp. 382–399). The constraints can be classified as natural (implied by natural laws of the modelled system) or practical (resulting from common sense, former practice, and placed to accelerate numerical searches). Initially, only natural constraints should be considered and, if possible, eliminated through appropriate transformations and

an introduction of slack variables — so that the nature of optima does not change. Any implicit constraint whose nature can be identified, should be mapped onto search domain D (see Atkinson and Donev, 1992, p. 189). Finally, only those practical constraints are added that are desirable either to maintain model validity or to speed up the computations. One has to be particularly careful while applying the latter ones — search space reduction may really accelerate working of the algorithm, but when applied over-eagerly, it can deprive the domain of the optimal solution altogether.

If the above procedure had been followed and implicit constraints were still active, then the application of a penalty function could be the solution. Various penalizing schemes are discussed in (Rao, 1996, pp. 487–517). The penalty function can, however, be applied only when one knows the extent to which the constraints are violated. If even that little data is unavailable, e.g. due to simulation failure, then the only approach suggested (see Walters, Jr. *et al.*, 1991, pp. 138–143) is to set the penalty function value to $+\infty$. Unfortunately, by doing that one makes the response surface like a sieve, with whom few solvers can cope.

To make matters worse, inconvenient response surfaces and constraints are frequently combined with moderate or even large dimensionality of the problem, thus rendering it intractable by direct search methods in their pure form. That is why so many authors (cf. Hammel, 1997, pp. 1–9 and most of the bibliography items cited above) insist on taking a deeper look into the model, and to overcome the difficulties by some kind of aggregation, mapping and heuristics.

Those suggestions are followed for most practical applications. For example, an automatic design procedure of a micropump is reported in (Meinzer *et al.*, 1996) where the costly finite element simulation was applied only to find the coefficients for some much faster mathematical model (computed by PSPICE circuit simulator), which was therefore explored by a genetic optimization algorithm. Next, (Brady and McGarvey, 1998) gives an account of a heuristic algorithm developed specifically to optimize the operating performance of a pharmaceutical laboratory. This algorithm uses a histogram of objective values, created by some standard direct search algorithm, as a starting point. Genetic algorithm, simulated annealing and tabu search were tested as standard algorithms. Another merger of genetic algorithm and tabu search features can be found in (Glover *et al.*, 1996). The authors present a technique called the scatter search, similar to genetic algorithm, with the difference that the offspring is created deterministically. An important supplement to the method is a neural network that makes predictions of the objective value, thus avoiding function evaluations at possibly bad points. The method was applied for optimal design of a job shop system.

Given the complexity of simulation optimization problems, parallel computing has created big hopes. Many of the routines mentioned in this paper can be executed in parallel, at least partially. Particularly all re-samplings of $f(\cdot)$ can be executed simultaneously; also the offspring generation in genetic algorithms can be done by many processors at the same time. Several algorithms have been developed specifically for the purpose of being executed in parallel. One of those (see Dennis and Torczon, 1991) is a simplex search routine, modified so that the simplex operations are performed with respect to the best vertice. Such a change requires not one but $\dim \mathbf{x}$ function evaluations per each single operation. One may also consider making some more steps in advance — this generates a lattice of points, and the evaluation of $f(\cdot)$ for them is performed in parallel. Another routine, described in (Hough and Kolda, 2001) and called parallel pattern search, consists of evaluating the

objective at several points in a neighbourhood of the current approximation $\mathbf{x}_n$, and choosing the best of them as $\mathbf{x}_{n+1}$. Those points, unlike in the case of simulated annealing, are chosen in a deterministic manner, thus forming the above mentioned pattern around $\mathbf{x}_n$. In general, the papers presenting parallel algorithms focus rather on their practical applicability, effectivity and tolerance to network faults rather than on strict mathematical proofs of their convergence.

3.4 Reality and the proposed hybrid approach

Two conclusions may be derived from the above survey of the literature. The first is that, given the problem specific features, direct search methods can only be applied. However, they are rarely used in their generic form; instead they are tuned to a concrete problem, and this tuning is based on a substantial knowledge about the system being optimized. The second conclusion is that parallel computing is commonly employed to mitigate large computational burden of the "lengthy simulation/direct search optimization" couple.

Experienced developers and researchers, asked in (Boesel *et al.*, 2001) to express their position statements as to the future of simulation optimization given the current state of the art, stressed that more effort should be put in making simulator and optimizer work together and not treat each other as a "black box" unaware of its partner capabilities and requirements. To attain this:

- simulation-optimization interface needs to be developed that will pass as much information in both directions; this is because
- the optimizer should react to the simulation in the midst of its run, e.g. to detect and cut early calculations unworthy for the optimization routine; good simulation-optimization interface makes it possible that
- the optimizer, equipped with problem classification procedure, will detect the nature of the optimization problem as presented by simulation output and will choose adaptively the optimization routine best suited for it; moreover
- whenever possible, the optimization should make the benefit of distributed and parallel computing to overcome routines high computational speed requirements.

Following the above postulates by managers and software developers is, undoubtedly, desirable but there are practical reasons that seriously jeopardize the proposed tight simulator-optimizer collaboration. Guilty parts are software users and developers. Unfortunately, there are more and more records of cases where the end user of a commercial simulation/optimization software *is not* interested in the nature of a system he/she wants to optimize. All one can and want do is to supply the optimization solver developer with an appropriate third-party simulator whose internal workings he does not know — and to request the working solution. Therefore, the developer is faced with an opaque piece of simulation software and a vague — if any — description of the nature of the simulated system. This also happens partially due to the policy of simulation software manufacturers, but prevalently due to more and more common attitude of customers: they are willing to pay, and in exchange they demand the working solution, without getting into details.

In author's opinion, such a situation, as unpalatable as it may seem, has to be finally accepted, and some general methodology has to be elaborated to handle the cases when

literally nothing more is known about the problem than the output objective values obtained from a black-box simulator. It is believed that employing some hybrid optimization scheme could be the right approach. Such a hybrid approach would be a mixture of several well known methods that has been improved to handle troublesome implicit constraints. Those methods would work sequentially, the solution obtained by one method being then the starting point for the next one. Moreover, such optimization scheme should make benefit from the parallel computing environment, if available.

Some initial steps towards this proposition were made, as reported in (Kamola and Malinowski, 2001). The authors describe the problem of searching for an optimal working point of a model of industrial power plant. Given the values of some design variables, the values of dependent variables are calculated by a simulator. The hybrid method was applied, consisting of CRS2 algorithm (see Price, 1987) followed by COMPLEX routine (see Box, 1965), the latter amended to support non-connected domains. The results obtained were satisfactory.

The current research carried by the author is directed towards the development of a general-purpose simulation optimization broker. Such broker allows to plug in a number of widely known optimization routines at one side, and a simulator (or a pool of simulators in the parallel case) at the other side. At the simulation side, the broker has to adapt to the amount of information available from the simulator, and to the possibilities to control the simulation process. At the optimization side, the broker has to adapt to the specifics of the routines used, invoking intelligently the most appropriate of them. It also has to handle the cases when the simulation fails. Comprehensive results of author's experience with simulation optimization practical problems, and of research on the criteria for switching between methods, are to be presented in dissertation *Algorithms for optimisation problems with implicit and ill-defined constraints*, now in preparation.

4 Summary

This paper was intended to present merely an overview of approaches for optimization problems where the objective function values can be obtained only by means of simulation. Such problems are mostly nonlinear, and gradient information is rarely available, so the attention was focused on direct search methods. This field of research, although divided initially in two areas of stochastic and deterministic optimization, appears to be equally populated with direct search methods. Those methods, possibly due to their innate distrust in function value evaluation at a single point, have gained their position both in the fields of global and stochastic problems.

Constraints still present a considerable difficulty in simulation-based optimization. Many authors suggest that they should be eliminated, which would require a deeper study on the model of the system being optimized. However, such study is often impossible, either because of model complexity, or because of software market reality.

The author proposes to accept the situation as it is, and to perform research on hybrid optimization methods that, with the help of parallel computing, can hopefully meet the customer needs.

References

Andradóttir S. (1995): *A Stochastic Approximation with Varying Bounds.* — Operations Research, Vol.43, No.6, pp.1037–1048.

Andradóttir S. (1996): *A Global Search Method for Discrete Stochastic Optimization.* — SIAM J. Optimization, Vol.6, No.2, pp.513–530.

Andradóttir S. (1998a): *Simulation Optimization*, In: Handbook of simulation: Principles, methodology, advances, applications, and practice (J. Banks, Ed.). — Chichester: John Wiley & Sons, Inc.

Andradóttir S. (1998b): *A Review of Simulation Optimization Techniques*, pp.151–158, In: Proceedings of the 1998 Winter Simulation Conference (D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, Eds.).

Atkinson A.C. and Donev A.N. (1992): *Optimum experimental design.* — Oxford: Clarendon Press.

Barretto M.R.P., Chwif L., Eldabi T. and Paul R.J. (1999): *Simulation Optimization with the Linear Move and Exchange Move Optimization Algorithm*, pp.806–811, In: Proceedings of the 1999 Winter Simulation Conference (P.A. Farrington, H.B. Nembhard, D.T. Sturrock and G.W. Evans, Eds.).

Barton R.R. and Ivey, Jr. J.S. (1991): *Modifications of the Nelder-Mead Simplex Method for Stochastic Simulation Response Optimization*, pp.945–953, In: Proceedings of the 1991 Winter Simulation Conference (B.L. Nelson, W.D. Kelton and G.M. Clark, Eds.).

Boesel J., Bowden, Jr. R.O., Glover F., Kelly J.P. and Westwig R. (2001): *Future of Simulation Optimization*, pp.1466–1469, In: Proceedings of the 2001 Winter Simulation Conference (B.A. Peters, J.S. Smith, D.J. Medeiros and M.W. Rohrer, Eds.).

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

Brady T. and McGarvey B. (1998): *Heuristic Optimization Using Computer Simulations: A Study of Staffing Levels in a Pharmaceutical Manufacturing Laboratory*, pp.1423–1428, In: Proceedings of the 1998 Winter Simulation Conference (D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, Eds.).

Bulgak A.A. and Sanders J.L. (1988): *Integrating a modified simulated annealing algorithm with the simulation of a manufacturing system to optimize buffer sizes in automatic assembly systems*, pp.684–690, In: Proceedings of the 1988 Winter Simulation Conference (M. Abrams, P. Haigh and J. Comfort, Eds.).

Carson Y. and Maria A. (1997): *Simulation Optimization: Methods and Applications*, pp.118–126, In: Proceedings of the 1997 Winter Simulation Conference (S. Andradóttir, K.J. Healy and B.L. Nelson, Eds.).

Chen H. and Zhu Y. (1986): *Stochastic Approximation Procedures with Randomly Varying Truncations.* — Scientia Sinica, Vol.29, No.9, pp.914–926.

Chong E.K.P. and Ramadge P.J. (1993): *Optimization of Queues Using an Infinitesimal Perturbation Analysis-Based Stochastic Algorithm with General Update Times.* — SIAM J. Control and Optimization, Vol.31, No.3, pp.698–732.

Delyon B. and Juditsky A. (1993): *Accelerated Stochastic Approximation.* — SIAM J. Optimization, Vol.3, No.4, pp.868–881.

Dennis J.E. and Torczon V. (1991): *Direct Search Methods on Parallel Machines.* — SIAM J. Optimization, Vol.1, No.4, pp.448–474.

Faccenda J.F. and Tenga R.F. (1992): *A Combined Simulation/Optimization Approach to Process Plant Design*, pp.1256–1261, In: Proceedings of the 1992 Winter Simulation Conference (J.J. Swain, D. Goldsman, R.C. Crain and J.R. Wilson, Eds.).

Findeisen W., Bailey F.N., Brdyś M., Malinowski K., Tatjewski P. and Woźniak A. (1980): *Control and Coordination in Hierarchical Systems.* — Chichester: John Wiley & Sons, Inc.

Fox B.L. and Heine G.W. (1995): *Probabilistic Search with Overrides.* — The Annals of Applied Probability, Vol.5, No.4, pp.1087–1094.

Fu M.C. (1990): *Convergence of a Stochastic Approximation Algorithm for the G1/G/1 Queue Using Infinitesimal Perturbation Analysis.* — Journal of Optimization Theory and Applications, Vol.65, No.1, pp.149–160.

Fu M.C. (1994): *A Tutorial Review of Techniques for Simulation Optimization*, pp.149–156, In: Proceedings of the 1994 Winter Simulation Conference (J.D. Tew, S. Manivannan, D.A. Sadowski and A.F. Seila, Eds.).

Gelfand S.B. and Mitter S.K. (1989): *Simulated Annealing with Noisy or Imprecise Energy Measurements.* — Journal of Optimization Theory and Applications, Vol.62, No.1, pp.49–62.

Gelfand S.B. and Mitter S.K. (1991): *Recursive Stochastic Algorithms for Global Optimization in* $\mathcal{R}^d$. — SIAM J. Control and Optimization, Vol.29, No.5, pp.999–1018.

Glover F., Kelly J.P. and Laguna M. (1996): *New Advances and Applications of Combining Simulation and Optimization*, pp.144–152, In: Proceedings of the 1996 Winter Simulation Conference (J.M. Charnes, D.J. Morrice, D.T. Brunner and J.J. Swain, Eds.).

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

Hough P.D. and Kolda T.G. (2001): *Asynchronous Parallel Pattern Search for Nonlinear Optimization.* — SIAM J. Scientific Computing, Vol.23, No.1, pp.134–156.

Humphrey D.G. and Wilson J.R. (1998): *A Revised Simplex Search Procedure for Stochastic Simulation Response-Surface Optimization*, pp.751–759, In: Proceedings of the 1998 Winter Simulation Conference (D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, Eds.).

Jacobson S.H. (1994): *Convergence Results for Harmonic Gradient Estimators.* — ORSA Journal on Computing, Vol.6, pp.381–397.

Kamola M. and Malinowski K. (2001): *Simulator-Optimizer Approach to Planning of Plant Operation; Ill-Defined Simulator Case*, In: A Proceedings volume from the IFAC Symposium, Patras Greece, 12-14 July 2000 (P.P. Groumpos and A.P. Tzes, Eds.).

Kamola M. and Miazga P. (2001): *Global and Local Optimization Algorithms in Automated Waveguide Design*, pp.97–105, In: Materiały V Krajowej Konferencji "Algorytmy Ewolucyjne i Optymalizacja Globalna". — Jastrzębia Góra.

Kleinman N.L., Hill S.D. and Ilenda V.A. (1998): *Simulation Optimization of Air Traffic Delay Cost*, pp.1177–1181, In: Proceedings of the 1998 Winter Simulation Conference (D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, Eds.).

Kushner H.J. and Yang J. (1993): *Stochastic Approximation with Averaging of the Iterates: Optimal Asymptotic Rate of Convergence for General Processes.* — SIAM J. Control and Optimization, Vol.31, No.4, pp.1045–1062.

Meinzer S., Quinte A., Gorges-Schleuter M., Jakob W. and SüßW. (1996): *Simulation and Design Optimization of Microsystem Based on Standard Simulators and Adaptive Search Techniques*, pp.322–327, In: Proceedings of European Design Automation Conference with EURO-VHDL'96 and exhibition on European Design Automation. — Geneva.

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

Pflug G.C. (1996): *Optimization of Stochastic Models.The Interface between Simulation and Optimization.* — Amsterdam: Kluwer Academic Publishers.

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

Polyak B.T. and Juditsky A.B. (1991): *Acceleration of Stochastic Approximation by Averaging.* — SIAM J. Control and Optimization, Vol.30, No.4, pp.838–855.

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

Rao S.S. (1996): *Engineering Optimization. Theory and Practice.* — John Wiley & Sons, Inc.

Robinson S.M. (1996): *Analysis of sample path optimization.* — Mathematics of Operations Research, Vol.21, pp.513–528.

Ruppert D. (1985): *A Newton-Raphson Version of the Multivariate Robbins-Monro Procedure.* — The Annals of Statistics, Vol.13, No.1, pp.236–245.

Shapiro A. and Wardi Y. (1996a): *Convergence Analysis of Gradient Descent Stochastic Algorithms.* — Journal of Optimization Theory and Applications, Vol.91, No.2, pp.439–454.

Shapiro A. and Wardi Y. (1996b): *Convergence Analysis of Stochastic Algorithms.* — Mathematics of Operations Research, Vol.21, No.3, pp.615–628.

Spall J.C. (1992): *Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation.* — IEEE Transactions on Automatic Control, Vol.37, No.3, pp.332–341.

Spendley W., Hext G.R. and Himsworth F.R. (1962): *Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation.* — Technometrics, Vol.4, No.4, pp.441–461.

Stuckman B., Evans G. and Mollaghasemi M. (1991): *Comparison of Global Search Methods for Design Optimization Using Simulation*, pp.937–944, In: Proceedings of the 1991 Winter Simulation Conference (B.L. Nelson, W.D. Kelton and G.M. Clark, Eds.).

Svensson A. (1997): *A Novel Simulation Based Optimization Method - Applied to the Dimensioning of Circuit Switched Networks*, In: Proceedings of the 1st World Congress on Systems Simulation. — Singapore.

Walters, Jr. F.H., Parker L.R., Morgan S.L. and Deming S.N. (1991): *Sequential simplex optimization: a technique for improving quality and productivity in research, development and manufacturing.* — CRC Press, Inc.

Wardi Y. (1990): *Stochastic Algorithms with Armijo Stepsizes for Minimization of Functions.* — Journal of Optimization Theory and Applications, Vol.64, No.2, pp.399–417.

Wei C.Z. (1987): *Multivariate Adaptive Stochastic Approximation.* — The Annals of Statistics, Vol.15, No.3, pp.1115–1130.

Yan D. and Mukai H. (1992): *Stochastic Discrete Optimization.* — SIAM J. Control and Optimization, Vol.30, No.3, pp.594–612.

Yan D. and Mukai H. (1993): *Optimization Algorithm with Probabilistic Estimation.* — Journal of Optimization Theory and Applications, Vol.79, No.2, pp.345–371.

Yin G. (1991): *On Extensions of Polyak's Averaging Approach to Stochastic Approximation.* — Stochastics and Stoschastics Reports, Vol.36, pp.245–264.