Struct Multidisc Optim
DOI 10.1007/s00158-013-0950-5

RESEARCH PAPER

Optimization under worst case constraints—a new global multimodel search procedure

Michael de Paly · Claudius M. Bürger · Peter Bayer

Received: 10 October 2011 / Revised: 5 April 2013 / Accepted: 30 May 2013

Abstract A new method is presented that combines heuristic global optimization and multi-model simulation for reliability based risk averse design. The so-called new stack ordering method is motivated from hydrogeology, where high-reliable groundwater management solutions are sought for with a demanding set of equally probable model alternatives. The idea is to only exploit a small subset of these model alternatives or realizations to approximate the objective function to reduce computational costs. The presented automatic procedure dynamically adjusts the subset online during the course of iterative optimization. The test with theoretical reliability based benchmark problems shows that the new method is efficient in regard to optimality and reliability of found solutions already with small subsets of all models. Compared with a previously presented first version of stack ordering, the presented generalized approach proves to be more robust, computationally efficient and of great potential for related problems in reliability based optimization and design. This conclusion is supported by the fact that the new variant requires about one fifth of the objective function evaluations of the older version in order to achieve the same level of reliability. We also show that these findings can be translated to real world problems by benchmarking the performance on a well capture problem.

Keywords Reliability optimization · Constraint optimization · Risk averse design

This work was presented in part at the Noisy Kriging-based Optimization (NKO) workshop, Bern, November 2010.

M. de Paly (✉)
Center for Bioinformatics Tübingen (ZBIT),
University of Tübingen,
Sand 1, 72076 Tübingen, Germany
e-mail: michael.depaly@uni-tuebingen.de

C. M. Bürger
Department of Geosciences,
University of Tübingen,
Sigwartstr.10, 72076 Tübingen, Germany

C. M. Bürger
Shell Global Solutions International B.V.,
Kessler Park 1, 2288 GS Rijswijk, The Netherlands

P. Bayer
Department of Earth Sciences, ETH Zurich,
Sonneggstr. 5, 8092 Zurich, Switzerland

1 Introduction

Computer simulations are common means to understand, characterize and predict real world phenomena. They are often a convenient, practical, and efficient alternative to real experiments. Behavior of materials, fluids, and technical circuits can be examined in a comparatively inexpensive way. In engineering-related disciplines simulation models are usually based on governing partial differential equations. These equations are derived from physical principles like conservation of energy, mass and momentum, and so-called constitutive relationships describing e.g. the response of a material to (external) forces. The model equations usually have to be solved numerically by iterative solution procedures, which makes computation time an important criterion for the applicability of a simulation-aided design procedure or so-called combined simulation-optimization (Azadivar 1999; Fu et al. 2005; Bayer et al. 2010). The latter involves an objective function that is evaluated based on the output of a model. An appropriate optimization algorithm typically suggests a sequence of solution alternatives, which all need to be evaluated to arrive at an optimal design.

Another advantage of the use of computer models is that the impact of variability or uncertainty in material properties and/or boundary conditions can be directly investigated by either some kind of uncertainty propagation (e.g. first or higher order second moment methods (Zhang and Lu 2004; Lu and Zhang 2005; Cirpka et al. 2004; Lu and Zhang 2007), polynomial chaos expansion (Ghanem and Spanos 1991; Debusschere et al. 2005; Lin and Tartakovsky 2009; Arnst et al. 2010) or by stochastic sampling through Monte-Carlo simulations (Binder 1979; Hubbard 2007). For real-world applications suitable uncertainty analysis is pertinent, since optimal technical solutions always need to encompass significant safety margins. However, both types of uncertainty assessment come at the expense of an even further increase of computational demand. Incorporating reliability estimation in combined simulation-optimization which results in double or even triple loops thus is computationally extremely demanding (Valdebenito and Schuëller 2010; Du et al. 2008). Thus efficient strategies that allow efficient reliability-based optimal design are needed. This work introduces a new approach in this field.

The development of the algorithm introduced here has its roots in hydrogeology. This discipline is concerned with the study of natural flow and transport of water and solutes in the subsurface. In this context, hydrogeologists are also involved in the risk averse design of groundwater remediation measures. What distinguishes a groundwater clean-up problem from a typical engineering problem is that it is not inherently a technical system that needs to be understood, but a natural one that can only be probed at comparatively few points. As a consequence, usually neither the exact location of the contamination source nor the exact flow and transport paths in the subsurface are known. The simulation of the impact of a technical measure, i.e. a groundwater remediation system, is therefore subject to considerable uncertainty. Fortunately, at least the general pattern of the flow path governing subsurface heterogeneity can be estimated by geostatistical simulation techniques utilizing hard geological information and soft knowledge.

While still not the standard in practice, simulation-optimization together with uncertainty assessment is an active area of research in hydrogeology. Probably the most common approach for hydrogeological uncertainty assessment are Monte-Carlo simulations based on the sampling of non-stationary, spatially-correlated random fields of subsurface parameters that are conditioned to measured data (Singh and Minsker 2008; Wagner and Gorelick 1989; Morgan et al. 1993). These realized parameter fields or—as referred to hereafter—realizations are an input for the numerical model. Each model outcome represents an equally-probable description of the subsurface flow and/or concentration field. Important for this study is the fact that the general conditioning of realizations to measured data requires an iterative inverse modeling procedure. Depending on the amount of conditioning data this usually is also a computation time consuming process, which is typically performed before the remediation system design question is approached. Hence, even though feasible, realization generation is usually not part of the design adaptation phase. Ultimately, a reliable remediation system has to be functional for all (conditional) realizations. This strict reliability requirement in combination with a discrete finite set of realizations for the random variables poses a challenge for most common reliability based optimization techniques. FORM (Hasofer and Lind 1974) or SORM (Tvedt 1990) based methods that require numerical optimization in the random variable space become cumbersome or even infeasible. On the other hand methods which base on Monte Carlo sampling and or estimation of the constraints probability density function (Xi et al. 2011) will most likely demand a full evaluation of the entire realization set in order to meet the required reliabilities. In practice, of course, there are computational limits on how many realizations can be evaluated for a particular remediation system setup. This is even more so, as the optimal setup is not known and subject to an optimization procedure. Mathematically we can call this a constrained optimization problem, where the decision variables are remediation system design parameters and the different (inequality) constraints are given through the evaluation of the functionality of a particular candidate solution for each realization. The numerical model can be required for both, the calculation of the objective function value and the evaluation of constraints. Due to model discretization, numerical approximation and non-smooth parameter fields, appropriate optimization algorithms need to be able to handle noisy, non-convex, nonlinear objective functions where gradient information is unreliable or not available. Given the sampling of a random field, the constraints are stochastic or noisy as well. Hence, heuristic or evolutionary algorithms are typically applied to these kinds of problems (Mantoglou and Kourakos 2007; Nicklow et al. 2010; Wu et al. 2006).

In the following, the method called new stack ordering is introduced, which represents an advanced and generalized version of a previously developed hydrogeological optimization method by Bayer et al. (2010). Then this method is applied to high reliability based benchmark problems in combination with two evolutionary algorithms. The optimization solutions are inspected for the obtained reliability and optimality, and special focus is set on the related computational effort. Afterwards a real world oriented well capture problem akin to the problem considered in Bayer et al. (2010) is chosen and the transferability of the findings of the theoretical benchmarks to a real world problem is evaluated.

2 Methods

2.1 Definition of the constraint optimization problem

In the following we will formulate this practical problem in formal mathematical terms. Let $\mathbf{x} \in \mathbb{R}^n$ be a vector containing $n \in \mathbb{N} \setminus \{0\}$ decision variables. For simplicity, let all components of $\mathbf{x}$ belong to compact subsets of $\mathbb{R}$ which define the n dimensional search space X of the optimization problem. We call $\mathbf{x}$ a point in the search space or a solution candidate. Let

$$h : \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}^{n_h}, (\mathbf{x}, r) \rightarrow h(\mathbf{x}, r) \tag{1}$$

denote the vector-valued simulation model output at n_h compliance points ($h(\mathbf{x}, r)$ is understood as a deterministic function neglecting possible machine dependent numerical round-off errors and n_h may be as large as the total number of model cells or nodes) for a particular solution candidate $\mathbf{x}$ and the realization r, where r is a uniquely defined identifier of a particular realization out of a set V of $n_V = |V|$ and $n_V \in \mathbb{N} \setminus \{0\}$ sampled realizations. These realizations are sampled once at the beginning of the optimization procedure and are kept for the entire optimization process.

Then the constraint i, with $i = 1, \ldots, n_h$ is fulfilled, if the model output at compliance point i, $h_i(\mathbf{x}, r)$, is below or equal to a pre-defined threshold $c_i \in \mathbb{R}$ and violated otherwise.Without loss of generality we thus define $h_i(\mathbf{x}, r) < 0$ for a constraint violation of solution candidate $\mathbf{x}$ for realization r.

The objective function is the total cost of a particular design $\mathbf{x}$ given as the scalar function $m : \mathbb{R}^n \rightarrow \mathbb{R}, \mathbf{x} \rightarrow m(\mathbf{x})$. Hence the optimization problem for a *single* realization r can be formulated as

$$\underset{\mathbf{x}}{\operatorname{argmin}} \, (m(\mathbf{x})) \tag{2}$$

subject to

$$\begin{aligned} h_1(\mathbf{x}, r) &\geq 0 \\ &\ldots \\ h_{n_h}(\mathbf{x}, r) &\geq 0 \end{aligned} \tag{3}$$

The final optimization goal is to achieve a pre-set target nominal reliability $R_t \in [0, 1)$. The nominal reliability can be calculated for every candidate solution $\mathbf{x}$ as follows: $R(\mathbf{x}) = \frac{n_r(\mathbf{x})}{n_V+1}$, where $n_r(\mathbf{x}) \in \mathbb{N}, n_r(\mathbf{x}) \leq n_V$ is the number of realizations for which no constraint is violated for this particular candidate solution $\mathbf{x}$. In this study we consider the worst case constraint. This means that for the given optimization problem *maximum nominal reliability* has to be achieved, i.e. any design $\mathbf{x}$ requires $n_r(\mathbf{x}) = n_V$ in order to achieve the pre-set target nominal reliability R_t:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \, (m(\mathbf{x})) \tag{4}$$

subject to

$$\begin{aligned} h_1(\mathbf{x}, r) &\geq 0 \\ &\ldots \\ h_{n_h}(\mathbf{x}, r) &\geq 0 \\ \text{for } r &= 1, \ldots, n_r = n_V \end{aligned} \tag{5}$$

2.2 Critical realizations

As formulated above, all n_V realizations have to be evaluated for a single candidate solution $\mathbf{x}$ in order to be sure that maximum nominal reliability is achieved. If, however, a constraint is violated for a particular candidate solution $\mathbf{x}$ for a single realization, one can be certain to have found an infeasible solution. A useful uncertainty assessment would require n_V to be as high as several hundreds or thousands for the evaluation of the nominal reliability of a particular $\mathbf{x}$. In practice, this is easily computationally too demanding, if the uncertainty assessment is part of an already computationally intense optimization procedure. Several ways have been suggested to approximate the calculation of nominal reliability and thereby saving computation time. One idea is the use of a so-called noisy-GA as proposed by Smalley et al. (2000), Singh and Minsker (2008) where only a small subset of realizations is selected randomly out of n_V available for the evaluation of a particular $\mathbf{x}$. A more structured approach was followed by Mantoglou and Kourakos (2007) and Kourakos and Mantoglou (2008) who tried to identify so-called critical realizations by an adaptive procedure during the search.

Put in simple terms, critical realizations are these realizations which have a predominant impact on the feasibility of a candidate solution $\mathbf{x}$. If a $\mathbf{x}$ is infeasible for a given realization it is also infeasible for at least one of the critical realizations. The effect of all realizations is therefore already accounted for by considering the critical realizations alone. More formally speaking, we define a set of critical realizations $V_X^{critical}$ given the search space X as follows:

Definition 1 The set of critical realizations $V_X^{critical}$ for search space X is one smallest subset of all realizations V with

$$\begin{aligned} \forall \mathbf{x} \in X \, \forall r \in V : \, &(\exists i \in \{1, \ldots, n_h\}, h_i(\mathbf{x}, r) < 0) \Rightarrow \\ &\left(\exists r^{critical} \in V_X^{critical} : h_i\left(\mathbf{x}, r^{critical}\right) < 0\right) \end{aligned} \tag{6}$$

This means if there is a realization $r \in V$ that is constraint violating at $\mathbf{x} \in X$ there is at least one $r^{critical} \in V_X^{critical}$ that is also constraint violating at $\mathbf{x}$. If this property holds for all $\mathbf{x}$ in the search space it is sufficient to only evaluate the constraint function for all $r^{critical}$ to achieve maximum nominal reliability.

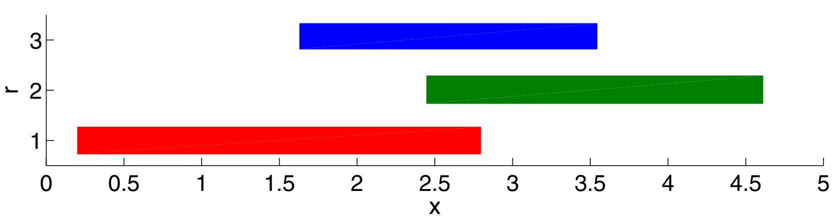

Fig. 1 Example for critical realizations. The range of **x** which is constraint violating for a given realization r_1, r_2, and r_3 marked by a *colored bar*

Figure 1 illustrates the concept of critical realizations with the help of a basic one-dimensional search space combined with three different realizations r_1, r_2, r_3. For each r the constraint violating ranges of $\mathbf{x} \in X$ are marked as colored bars. In this example, the minimal set of critical realizations for the entire search space $X = [0, 5]$ contains realizations r_1 and r_2. The idea of critical realizations can also be applied to subspaces of the whole search space. For instance, if the considered search space for **x** is restricted to $X = [2, 3]$ in the example the corresponding set of critical realizations $V_{[2,3]}^{critical}$ would only consist of realization $r^{critical} = 3$.

If $\left|V_X^{critical}\right|$ is small—this is indicated by the empirical results obtained by Bayer et al. (2004, 2010) and theoretically plausible for the hydrogeological problem stated above—candidate solutions with maximum nominal reliability can be found with considerable computational savings by evaluating only critical realizations instead of all n_V realizations.

Finding $V_X^{critical}$ can be seen as solving a set cover problem. To each r we can assign the subset $X_r^{violating} := \{\mathbf{x} \in X,\ \exists i \in \{1, \ldots, n_h\} :\ h_i(\mathbf{x}, r) < 0\}$ where it is constraint violating. $V_X^{critical}$ then contains all r required to cover the entire constraint violating $X^{violating} := \cup_{r=1}^{n_V} X_r^{violating}$ subset of X. Unfortunately solving the set cover problem itself is NP-hard and above all requires complete information about which elements belong to each subset. This means for each r the constraint violating sub spaces of $X_r^{violating}$ have to be known a-priori. Similar to the work of Mantoglou and Kourakos (2007), Bayer et al. (2004, 2010) try to alleviate the problems mentioned above by a greedy strategy for finding potentially critical realizations during the actual optimization process. In this study, a new theoretically underpinned version of this greedy strategy called 'stack ordering' is proposed.

The challenge to identify critical realizations from a finite set is also one aspect that distinguishes the 'stack ordering' method from the approaches of Avigad and Branke (2008) and Avigad and Coello (2010) who try to achieve worst case (i.e. maximum nominal) reliability by a two-criteria or multi-objective optimization procedure. The later aim at identifying the pareto-front of the cost function and a real-valued continuous constraint violation function. For the presented hydrogeological problem (and likely for analogous cases from other disciplines), however, the constraint space can not be searched in a continuous fashion. Due to the required inverse modeling to condition the underlying random parameter field to measured data it is virtually impossible to circumvent a Monte-Carlo sampling procedure for realistic applications. Because of the random sampling the set of realizations represented by a finite (but large) number of integers does not come with a distance - objective function value correlation a-priori (i.e. two close integers may not at all represent similar constraints). Therefore the incorporation of constraint violation as a second optimization criterion will result in a cumbersome mixed-integer problem which very likely resembles a needle in the haystack problem.

2.3 Stack ordering

The new stack ordering approach presented in this paper is based on the concept developed by Bayer et al. (2010), which in the following is called classic stack ordering (cSO). The basic idea of cSO is to speed up the constraint evaluation process by evaluating only a small number of at most S_{eval} considered critical realizations for each candidate solution. This set of critical realizations is approximated by a heuristic which is outlined in Fig. 2 and described in more detail in the following paragraphs.

cSO has to be coupled with an optimization algorithm that generates a single candidate solution $\mathbf{x}^{(t)}$ at each step t of the optimization process beginning with $\mathbf{x}^{(0)}$. In the case of a population based optimization algorithm the candidate solutions **x** within the population are evaluated sequentially. The evaluation of each solution candidate is considered to be an optimization step t.

At the start of the optimization process, cSO is initialized with:

1. a set V of realizations $r \in V$,
2. a maximum number of realizations to be evaluated per candidate solution S_{eval},
3. a fixed decay factor $k \in (0, 1)$.

The realizations are generated only once before the optimization procedure. In order to ensure a representative stack the number of realizations has to be sufficiently large and

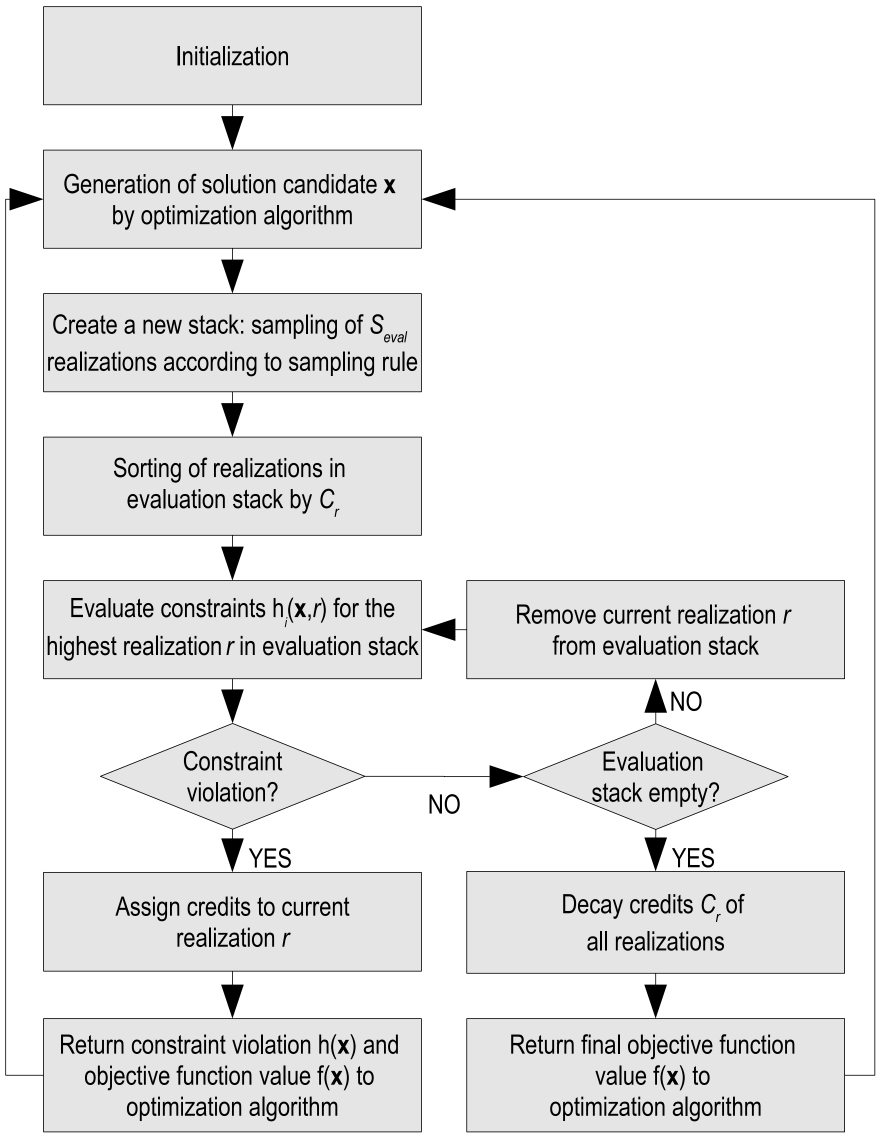
Initialization
Generation of solution candidate **x** by optimization algorithm
Create a new stack: sampling of S_{eval} realizations according to sampling rule
Sorting of realizations in evaluation stack by C_r
Evaluate constraints $h_i(\mathbf{x},r)$ for the highest realization r in evaluation stack
Remove current realization r from evaluation stack
NO
Constraint violation?
NO
Evaluation stack empty?
YES
YES
Assign credits to current realization r
Decay credits C_r of all realizations
Return constraint violation h(**x**) and objective function value f(**x**) to optimization algorithm
Return final objective function value f(**x**) to optimization algorithm

Fig. 2 Flow chart of the classic stack ordering algorithm

all realization have to be equally likely. Additionally, each realization r is assigned a real number $C^{(t)}(r)$, the so-called credit. $C^{(t)}(r)$ varies over t and is initialized to 0, i.e. $\forall r$: $C^{(0)}(r) = 0$.

At each optimization step t cSO samples a subset of realizations called evaluation stack $V_S^{(t)}$ of fixed size $\forall t$: $\left|V_S^{(t)}\right| = S_{eval}$. $V_S^{(t)}$ contains the realizations that will be used for constraint evaluation of the candidate solution $\mathbf{x}^{(t)}$. The evaluation stack $V_S^{(t)}$ is sampled based on the credits as follows:

1. All realizations with a credit

$$C^{(t)}(r) > \ln(S_{eval}) \tag{7}$$

are directly copied to the front positions of $V_S^{(t)}$ (if more than S_{eval} such realizations exist, the filling is stopped after a total number of S_{eval} realizations are assigned to the evaluation stack).

2. If there are still positions to be filled, the remaining positions up to S_{eval} are filled by probabilistic sampling of not yet selected realizations in V. Using realizations with higher credits first the realizations are sampled without replacement with the following probability for evaluation stack inclusion:

$$\mathrm{Prob}^{(t)}(r) = \frac{C^{(t)}(r) + 1}{\ln(S_{eval}) + 1} \tag{8}$$

After the sampling of $V_S^{(t)}$ the current candidate solution $\mathbf{x}^{(t)}$ will be evaluated for the realizations in the evaluation stack in ascending order of their position in the current evaluation stack. This position is given by sorting the realizations in the evaluation stack by their current credits $C^{(t)}(r)$ in descending order, i.e. the realization with the highest credit is at position 1, the realization with the second highest credit is at position 2, etc. assuming all credits are pairwise distinguishable.

If all S_{eval} realizations in $V_S^{(t)}$ are evaluated and no constraint violation occurred, $\mathbf{x}^{(t)}$ is considered to be feasible. In this case, the credit of all realizations is reduced by a fixed decay factor k as in:

$$\forall r \ : C^{(t+1)}(r) = (1-k)C^{(t)}(r) \tag{9}$$

Afterwards the objective function value $m\left(\mathbf{x}^{(t)}\right)$ is returned to the optimization algorithm which then continues with the next optimization step $t+1$.

Otherwise the first (in the order of evaluation stack processing) realization r^* that results in a constraint violation, i.e. $\exists i \in \{1,\ldots,n_h\}$ such that $h_i(\mathbf{x}, r^*) < 0$, will be assigned a new credit $C^{(t+1)}(r^*)$ for the next optimization step based on its position $\text{pos}^{(t)}(r^*)$ in the evaluation stack according to:

$$C^{(t+1)}(r^*) = C^{(t)}(r^*) + \ln\left(\text{pos}^{(t)}(r^*)\right) \tag{10}$$

All other realizations $r \in V_t^S \setminus \{r^*\}$ are set to

$$C^{(t+1)}(r) = C^{(t)}(r) \tag{11}$$

Then the evaluation of the stack stops and an approximate objective function value as well as the fact of a constraint violation is returned to the optimization algorithm. The algorithm continues with optimization step $t+1$.

The only two parameters of cSO are the evaluation stack size S_{eval} and the decay factor k. The latter is supposed to limit the duration of formerly critical realizations in the evaluation stack as the search progresses. The rationale for this follows the understanding that the set of critical realizations is dependent on the current considered range of the search space. The evaluation stack size S_{eval} on the one hand ensures computational savings, since only a small subset of V is evaluated even for a feasible candidate solution. On the other hand S_{eval} should ideally contain $V^{critical}_{[\mathbf{x}^{(opt)}-\epsilon,\mathbf{x}^{(opt)}+\epsilon]}$ (where ϵ is a vector of the same dimension as $\mathbf{x}$ with all components equal to the same (small) positive real number) arround the feasible global optimizer $\mathbf{x}^{(opt)}$ at the end of the search, so that it should not be smaller than the expected number of critical realizations.

2.4 New stack ordering

Based on the properties of the set of critical realizations $V_X^{critical}$ given in Section 2.2 we propose an improved greedy strategy for estimating $V_X^{critical}$. A suitable strategy for solving the set cover problem is to iteratively expand the preliminary set of chosen subsets e.g. realization $\hat{V}_X^{critical}$ by the subset which reduces the cardinality of yet uncovered elements the most. In the context of continuous constraint optimization this is the realization r which has the highest volume of constraint violating $x_r^{violating} \in X_r^{violating}$ which are not yet covered by the preliminary set of critical realizations $X_r^{violating} \setminus X^{violating}_{\hat{V}_X^{critical}}$. Here we have defined $X_W^{violating}$ with W being a set of n_W realizations indexed by $l = 1,\ldots,n_W$ as $X_W^{violating} := \cup_{l=1}^{n_W} X_l^{violating}$. Since $X_r^{violating}$ is not a-priori known for a given r we propose to approximate the ranking of the realizations r by their constraint violating probability $p_{\mathbf{x},r}$ estimated from observed samples during the optimization.

2.4.1 *Estimating the probability of constraint violation*

A constraint violation is expressed by the deterministic indicator function $c(\mathbf{x}, r)$ which becomes 1 if any constraint $h_i(\mathbf{x}, r)$ is violated by realization r for the solution candidate $\mathbf{x}$ and is 0 otherwise.

$$c(\mathbf{x}, r) = \begin{cases} 1 & \text{if} \quad \exists i \in \{1,\ldots,n_h\} : h_i(\mathbf{x}, r) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

However, we do not know the outcome of $c(\mathbf{x}, r)$ before the sampling of $\mathbf{x}$ is performed by the overlying search algorithm. Therefore $\mathbf{x}$ can be considered as a random variable from the perspective of the stack ordering algorithm. This makes the constraint evaluation for a given realization r a stochastic process, for which we in every step of the optimization process observe whether a constraint violation has occurred or not. Based on this, the central assumption of the new stack ordering algorithm is that the occurrence of a constraint violation within the search for a realization r can be considered as a Bernoulli process:

Assumption 1 *For any (possibly finite) sequence $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, ... of positions $\mathbf{x}$ in the search space X and realizations $r \in V$ the value of the function $c\left(\mathbf{x}^{(t)}, r\right)$ $(t = 1, 2, \ldots)$ is considered to be a binary random variable such that for each r the whole sequence can be treated as a Bernoulli process where $c(\mathbf{x}, r)$ is Bernoulli distributed with the success probability p_r, which denotes the probability of a constraint violation.*

In the new stack ordering framework we order the stack according to decreasing probability of a constraint violation p_r, i.e. stack position one holds the realization with the highest p_r. In order to do so, we have to obtain an estimate for p_r for each realization. Based on the probability density function for $x \ \ p_x$, the conditional distribution of a constraint violation given $x \ \ p_{r|x}$ one can obtain the probability of a constraint violation for a particular realization through marginalization over $\mathbf{x}$:

$$p_r = \int_{\mathbf{x}} p_{x,r}(\mathbf{x}, r) d\mathbf{x} = \int_{\mathbf{x}} p_{r|x}(r|\mathbf{x}) p_x(\mathbf{x}) d\mathbf{x} \tag{13}$$

where $p_{x,r}$ is the joint constraint violation distribution for x, r. This relationship allows us to determine p_r from the observation of constraint violations caused by a realization r for the solution candidates $\mathbf{x}$ that are generated during the optimization process without a-priorily knowing $\mathbf{x}$.

Therefore we can reduce the problem to a parameter estimation problem where we have to estimate the success probability $p = p_r$ of a Bernoulli process based on the observations whether an evaluated realization is constraint violating or not indicated by $c(\mathbf{x}, r)$.

Using the likelihood function:

$$p_r \mapsto p(c(\mathbf{x}, r)|p_r) \tag{14}$$

the posterior probability $p(p_r|c(\mathbf{x}, r))$ for p_r can be expressed as:

$$p(p_r|c(\mathbf{x}, r)) = \frac{p(c(\mathbf{x}, r)|p_r)p(p_r)}{\int p(c(\mathbf{x}, r)|p_r)p(p_r)dp_r} \tag{15}$$

This expression can be conveniently solved using a conjugate prior for p_r which has an identical algebraic form for prior $p(p_r)$ and posterior $p(p_r|c(\mathbf{x}, r))$.

For a Bernoulli process the conjugate prior is a Beta distribution (Fink 1997) with density:

$$f_{Beta}(p) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1 - p)^{\beta-1} \tag{16}$$

The hyper parameters of the prior are α_p and β_p with their corresponding a-posterior parameters:

$$\alpha_r^{(t+1)} = \alpha_p^{(t)} + c_r^{(t)} \tag{17}$$

$$\beta_r^{(t+1)} = \beta_p^{(t)} + n_r^{(t)} - c_r^{(t)} \tag{18}$$

where $n_r^{(t)}$ is the number of all constraint evaluations for realization r in the course of the entire optimization process (up to iteration (t)) and $c_r^{(t)} = \sum_{l=1}^{n_r^{(t)}} c(\mathbf{x}^{(t_l)}, r)$ (with (t_l) only indexing those elements of the Bernoulli sequence where c was evaluated for the particular realization r) is the number of the actual observed constraint violations for this realization.

Using the mean of the resulting beta distribution

$$p_r^{(t+1)} = \frac{\alpha_r^{(t+1)}}{\alpha_r^{(t+1)} + \beta_r^{(t+1)}} \tag{19}$$

we can now assign an expected probability $p_r^{(t+1)}$ of a constraint violation to each realization r.

Of course, the introduction of a Bayesian approach in combination with Assumption 1 requires some discussion. For once, it is unknown what kind of a distribution the optimization algorithm samples. Furthermore, the success probability p of the Bernoulli process will depend on the position $\mathbf{x}$ within the search space. Hence we implicitely assume a stationary Bernoulli process where in the application case there is none. Accordingly, Bayesian updates will only be correct or of reasonable usefulness in a certain neighborhood of $\mathbf{x}$ where the Bernoulli process is stationary. However, as will be shown by the benchmark cases, the advantage of adopting the Bayesian approach lies in the increased mobility of the realizations at the front positions of the stack. Compared to cSO, highly credited realizations can much more easily be removed from their dominating positions, when the search moves to new positions in the search space. In addition to that Bayesian updating appears to be so effective that increased mobility in the stack does not deteriorate the critical-realization-identifying property of the stack.

The Bayesian approach as discussed in this section also assumes the observations for each realization to be independent from other realizations. This independence of observations can not be guaranteed if the evaluation of the realization is stopped early if a constraint violation occurred or after S_{eval} realizations have been evaluated. The following section elaborates further on this.

2.4.2 Algorithmic framework of the new stack ordering algorithm

The proposed new stack ordering algorithm, nSO, uses the same general structure as the cSO (Fig. 2) but greatly differs in the way how realizations are sampled into the evaluation stack $V_S^{(t)}$. Instead of a credit value C_r each realization is assigned the number of constraint evaluations n_r and the number of constraint violations c_r for realization r which have been encountered so far in the optimization process.

The new stack ordering sorts all realizations r by their expected probability of a constraint violation p_r which is estimated using the procedure given in Section 2.4.1. The realizations are then evaluated deterministically starting with the realization with the highest p_r until a constraint violating realization r^* is found or S_{eval} realizations are evaluated. For each evaluated realization r the number of function evaluations n_r and the number of observed constraint violations c_r are then updated accordingly. A constraint violation is reported to the overlying optimization algorithm in the same manner as in the cSO scheme, if one of the evaluated realizations resulted in a constraint violation. If none of the evaluated S_{eval} realizations has caused a constraint violation for the given $\mathbf{x}$, this candidate solution is believed to be feasible.

Since the evaluation of the stack stops as soon as a constraint violation occurs, the estimated p_r based on n_r and c_r do not correctly reflect the true constraint violation probabilities. Instead they reflect the probability of a constraint violation within the volume of X which is not yet covered by a higher ranked realization in the evaluation stack. This makes the estimated p_r more akin to the normalized volume of constrained solutions that will be additionally covered if r is included into the set of critical realizations. The later is required by the aforementioned greedy strategy for solving the set covering.

2.4.3 Decay of n_r and c_r

The proposed scheme provides correct estimates for p_r under the assumption that the success probability of the Bernoulli process for c is independent of $\mathbf{x}$. This stationarity assumption does not hold in general. Therefore it may be beneficial to decrease the impact of older samples on the estimation of p_r. This can be done in a similar fashion as the decay of C_r in cSO. In the new stack ordering scheme the recorded number of function evaluations $n_r^{(t)}$ and constraint violations $c_r^{(t)}$ for each realization decay by:

$$n_r^{(t)'} = (1-k)n_r^{(t)} \qquad c_r^{(t)'} = (1-k)c_r^{(t)} \tag{20}$$

As a consequence $n_r^{(t)'}$ and $c_r^{(t)'}$ are used in (17) and (18). In contrast to the cSO the decay is employed once at each time step regardless of the feasibility of $\mathbf{x}$ resulting in a more stable behaviour.

3 Performance evaluation

3.1 Benchmark functions

In order to evaluate the performance of the stack ordering algorithm it has to be combined with an optimization algorithm that iteratively solves a constrained benchmark function. As the base objective function we have for all evaluation runs chosen the hypersphere function, as given in:

$$m(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{21}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ with ^{T} denoting the transpose. The simplicity of $m(\mathbf{x})$ minimizes the impact of the optimization algorithm and its performance on the benchmark and instead allows to focus on the performance of the stack ordering procedure.

3.2 Benchmark constraints

The main application area of stack ordering is the efficient optimization of stochastic constraint functions where the stochastic properties of the constraints are expressed by a set of equally likely realizations. We define three stochastic constraints of differing complexity, which can be handled analytically. Because of their low computational cost they allow a more thorough benchmarking than using real world application models which often have high computational demand. In contrast to more realistic examples from reliability based design practice the selected problems also are more perspicuous than black box numerical models and generally conceivable. The three benchmark constraints $\left({}^1h, {}^2h, {}^3h\right)$ are given in Table 1. Note that these constraints are no longer given for a finite number of compliance points as defined earlier, but describe manifolds of the search space. A constraint violation occurs, if the current point or position in the search space $\mathbf{x}^{(t)}$ is also a point of the manifold. Hence, the index i of $h_i(\mathbf{x}, r)$ is no longer necessary. More technically speaking, each of the three benchmark contraints is defined by a constraint function as stated in the second column of Table 1 and a set of realizations. Thereby the index j refers to the jth sample from the uniformly sampled range given in the third column of the table and $v^{(j)}$ denotes the obtained sample value. In case of ${}^2h$ and ${}^3h$ two- and three-dimensional vectors are sampled and we denote the individual components as $v_1^{(j)}$, $v_2^{(j)}$ and $v_3^{(j)}$, and if ranges apply to all components by $v_{1,2}^{(j)}$ or $v_{1,2,3}^{(j)}$ accordingly. The number of sampled realizations depends on the benchmark and is stated in the rightmost column. We also introduce the notation $x_{>i} = k$ with $\mathbf{x} \in \mathbb{R}^n$ and $i \in \mathbb{N} \setminus \{0\}$, $k \in \mathbb{R}$ to denote that all componets of $\mathbf{x}$ with an index larger than i are equal to k. The following paragraphs provide a short description of the benchmark constraints and their essential properties.

3.2.1 Constraint ${}^1h$

${}^1h$ is a linear constraint which for each realization bisects the search space into a feasible and an infeasible region

Table 1 Definition of the benchmark constraint functions

	Constraint definition	Realization range	No. of realizations
${}^1h(\mathbf{x}, j)$	$x_1 - v^{(j)} \geq 0$	$0 \leq v^{(j)} \leq 1$	1000
${}^2h(\mathbf{x}, j)$	$\left(x_1 - v_1^{(j)}\right)^2 \cdot \left(x_2 - v_2^{(j)}\right)^2 \geq 0.1$	$-0.25 \leq v_{1,2}^{(j)} \leq 0.25$	900
${}^3h(\mathbf{x}, j)$	$\sum_{i=1}^{3}\left(\left(x_i - v_i^{(j)}\right)^2 - 10\cos\left(2\pi\left(x_i - v_i^{(j)}\right)\right)\right) \geq 0$	$-0.26 \leq v_{1,2,3}^{(j)} \leq 0.26$	27000

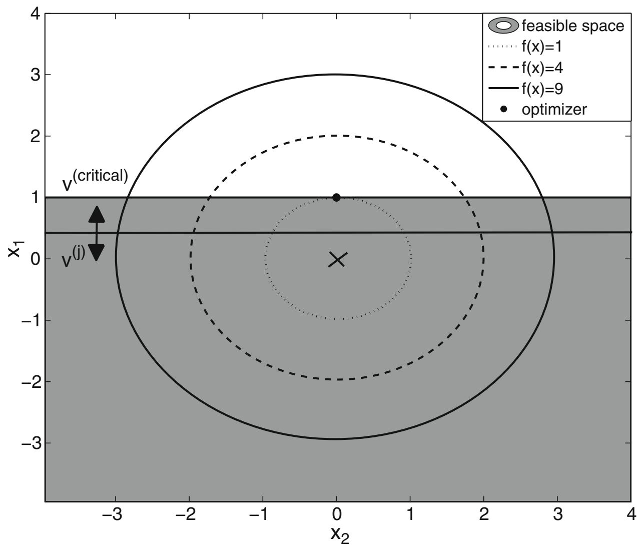

Fig. 3 Projection of ^1h into the (x_1,x_2) plane

along the hyperplane $x_1 = v^{(j)}$. Since all hyperplanes share the same normal vector, there is only one critical realization with $v^{(critical)} = \arg\max\left(v^{(j)}\right)$ $(j = 1,\ldots,1000)$. Figure 3 shows a projection of the (x_1,x_2)-plane of the search space and the feasible region. The resulting feasible global optimizer $\mathbf{x}^{opt}$ is located at $\left(x_1^{opt} = v^{(critical)}; x_{>1}^{opt} = 0\right)$ with $m\left(\mathbf{x}^{opt}\right) = \left(v^{(critical)}\right)^2$. With 1000 realizations and a dimensionality of one, ^1h has the highest resolution of realizations of all evaluated benchmarks.

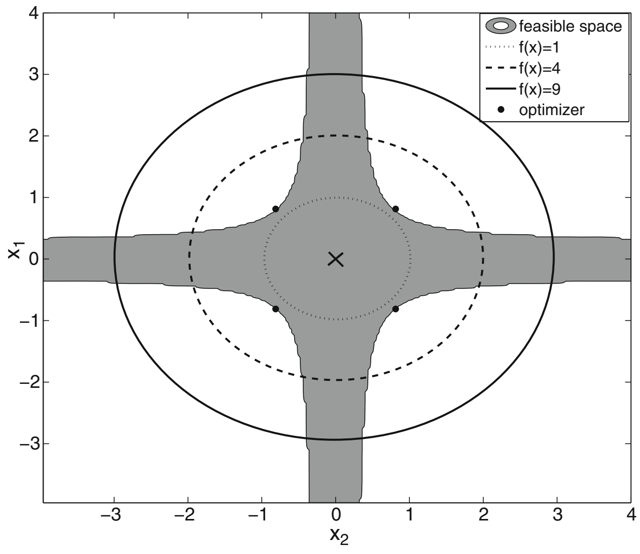

Fig. 4 Projection of ^2h

3.2.2 Constraint ^2h

The second benchmark constraint ^2h is more complex than ^1h because of its non-linearity and non-convexity. As shown in Fig. 4, ^2h is two dimensional and symmetric with respect to the coordinate axes x_1 and x_2. Under the assumption of infinitely many realizations v within the ranges given in Table 1, the symmetry results in four valid global optimizers at

$$\begin{aligned}
(x_1^{opt} &= d, x_2^{opt} = d, x_{>2}^{opt} = 0)\\
(x_1^{opt} &= d, x_2^{opt} = -d, x_{>2}^{opt} = 0)\\
(x_1^{opt} &= -d, x_2^{opt} = d, x_{>2}^{opt} = 0)\\
(x_1^{opt} &= -d, x_2^{opt} = -d, x_{>2}^{opt} = 0)\\
&\text{with } d = \sqrt[4]{0.1} + 0.25.
\end{aligned}$$

These global optimizers have an objective function value of

$$m(\mathbf{x}) = 2 \cdot \left(\sqrt[4]{0.1} + 0.25\right)^2.$$

Other than ^1h, ^2h has a large set of critical realizations $V_X^{critical}$ which depends heavily on the considered region of the search space X. However the cardinality of the critical set of the epsilon region around the global optimizers $V^{critical}_{[\mathbf{x}^{opt}-\epsilon,\mathbf{x}^{opt}+\epsilon]}$ grows only slowly with increasing ϵ. Although being 2-dimensional ^2h has only 900 realizations resulting in a much coarser covering of the constraint space when compared to ^1h. Such a coarse covering is typical for many real world problems where the sampling of realizations is often expensive and sampling of a larger number of realizations becomes prohibitively costly (Bayer et al. 2004; Cirpka et al. 2004; Mantoglou and Kourakos 2007; Wu et al. 2006).

3.2.3 Constraint ^3h

The third benchmark constraint is based on a 3-dimensional Rastrigin function making it the most complex constraint function used in this study. Under the assumption of maximum nominal reliability this constraint results in a disjoint feasible space with a large number of feasible local optima. Similar to ^2h, ^3h is symmetric with respect to the axes x_1, x_2, and x_3. This results in 24 global optimizers which are approximately located at

$$\begin{aligned}
&(x_1 = \pm 0.5042, x_2 = \pm 0.5042, x_3 = \pm 1.4965, x_{>3} = 0)\\
&(x_1 = \pm 0.5042, x_2 = \pm 1.4965, x_3 = \pm 0.5042, x_{>3} = 0)\\
&(x_1 = \pm 1.4965, x_2 = \pm 0.5042, x_3 = \pm 0.5042, x_{>3} = 0).
\end{aligned}$$

Additionally eight misleading local optimizers that act as traps are located at

$$(x_1 = \pm 0.5042, x_2 = \pm 0.5042, x_3 = \pm 0.5042, x_{>3} = 0).$$

These are governed by an extremely low number of constraint violating realizations which are hard to identify. With 27, 000 realizations this benchmark has the by far highest number of realizations. However, because of its larger dimensionality the effective resolution of realizations is much lower than for constraint ${}^{1}h$. Figure 5 shows a projection of an octant of ${}^{3}h$. Zero is located at the upper left corner. Areas with 100 % reliability are marked in dark blue and are enclosed by areas with 99 % reliability that are marked in light grey. The positions of the three visible feasible global optimizers are indicated by green stems.

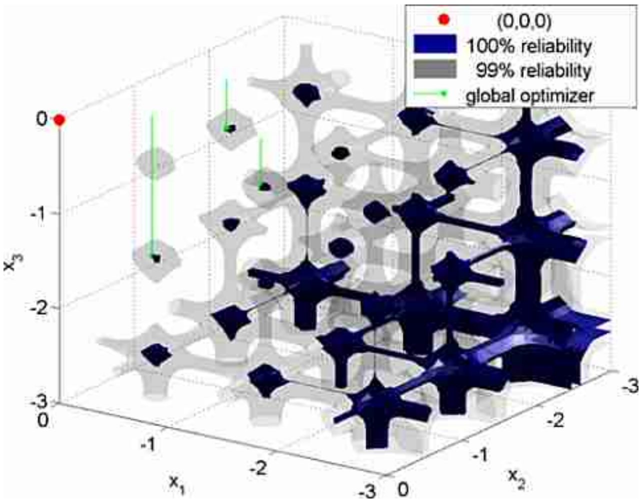

Fig. 5 Projection of ${}^{3}h$

3.3 A real world benchmark—the optimal well capture problem

In addition to the theoretical benchmark problems we also consider a real world oriented application case akin to the well capture problem described in Bayer et al. (2010). Here a numerical model simulates the movement of contaminants in groundwater. The contaminants have to be reliably captured by a well. The pumping rate of this well is ideally kept small to minimize the operational costs. Even for a given hydraulic regime it is a challenge to identify that location for the well, where the contaminants can be captured with lowest pumping (Bayer et al. 2008). The main reason is that the natural flow field commonly is non-uniform because of the heterogeneous spatial distribution of hydraulic conductivity. This is typically accounted for in the numerical groundwater model by varying conductivity values in the numerical grid cells. As typically measurements are only available for a few locations, assigning hydraulic conductivity values to the numerical grid is usually carried out by geostatistical interpolation or simulation techniques. The uncertainty in the interpolated hydraulic conductivity is accounted for by generating a set of equally probably hydraulic conductivity realizations. Each of these realizations results in a different flow field. A reliable well configuration guarantees capture for a high number or all of the realizations.

The application case, numerical model settings as well as the geostatistical assumptions for the generation of the hydraulic conductivity realizations are described in Bayer et al. (2010), and the reader is referred to that work for more details. In short, a hypothetical heterogeneous hydraulic conductivity distribution is assumed for an area of 100 m×150 m, for which only at 40 positions the true values are known. The latter are taken from a virtual reality, that is, one particular hydraulic conductivity field/distribution which is understood as an (unknown) real world condition. Contaminant flow paths are simulated by particle tracking. Particles are released into the aquifer from a permanent source, and the task is to find the optimal well placement in a pre-determined area (50 m×50 m) downgradient of the source zone, which is pre-determined and covers 50 m in both coordinate directions. In order to evaluate the quality of a solution candidate, which consists of the position of the well $(w_x;w_y)$ and the pumping rate w_p, the flow conditions for a realization are simulated by the program MODFLOW (Harbaugh et al. 2000). Contaminant propagation and potential capture is modeled using MODPATH (Pollock 1994). With the goal of minimizing the pumping rate the objective function becomes:

$$ {}^{w}m = w_p \qquad x = (w_x, w_y, w_p) \tag{22}$$

with the constraint of optimal contaminant capture:

$$ {}^{w}h(\mathbf{x}, j) = 0 \tag{23}$$

where ${}^{w}h(\mathbf{x}, j)$ is the number of uncaptured particles for solution candidate $\mathbf{x}$ at realization j.

3.4 Stack ordering parameters

3.4.1 Prior

The choice of the prior (α_p, β_p) plays an essential role for the performance of the proposed algorithm. The most commonly chosen prior in the absence of information about the underlying distribution is the Jeffreys prior (Jeffreys 1946) which for the Bernoulli distribution is:

$$\alpha_p = \beta_p = 1/2 \tag{24}$$

In the following the new stack ordering algorithm is abbreviated jSO when used with Jeffreys prior.

Alternatively we also investigated an improper pessimistic prior with

$$\alpha_p = 1 \qquad \beta_p = 0. \tag{25}$$

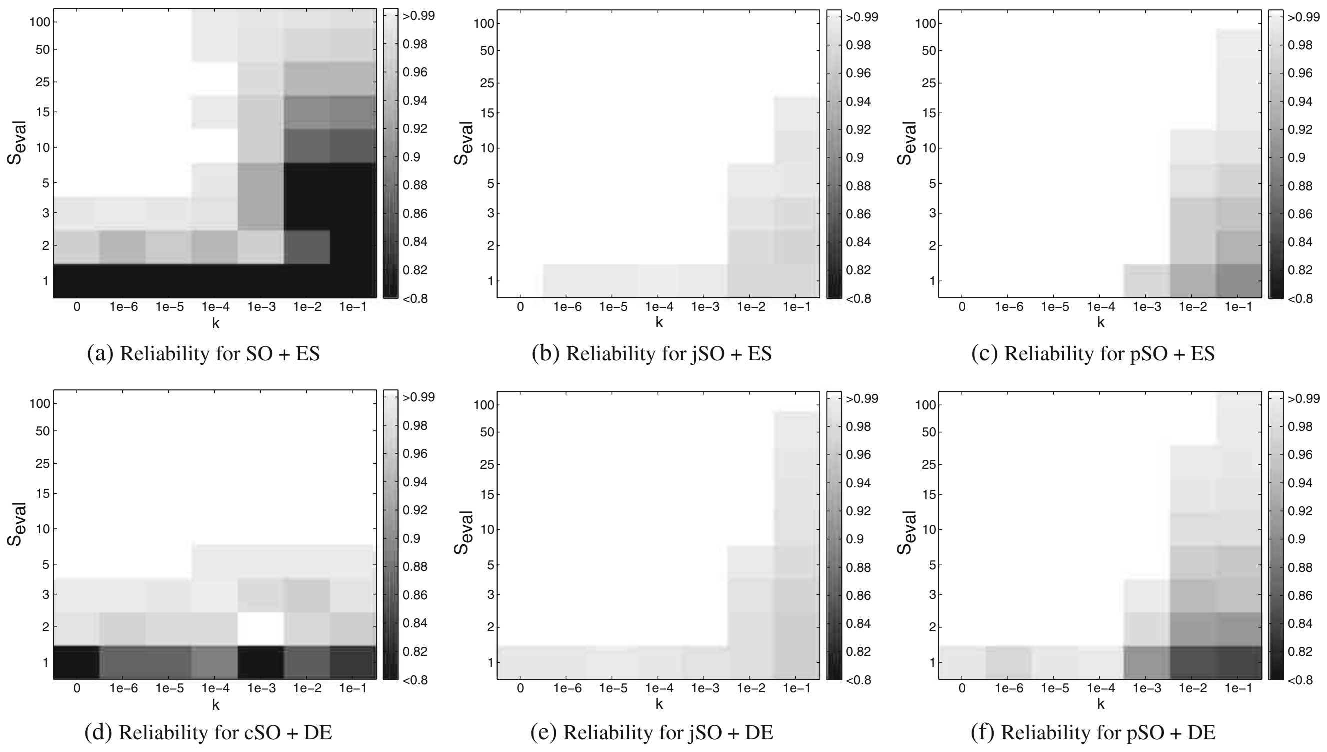

Fig. 6 Dependency of the achieved nominal reliability on S_{eval} and k for cSO, jSO, and pSO in combination with CMA-ES and DE on ${}^1h$

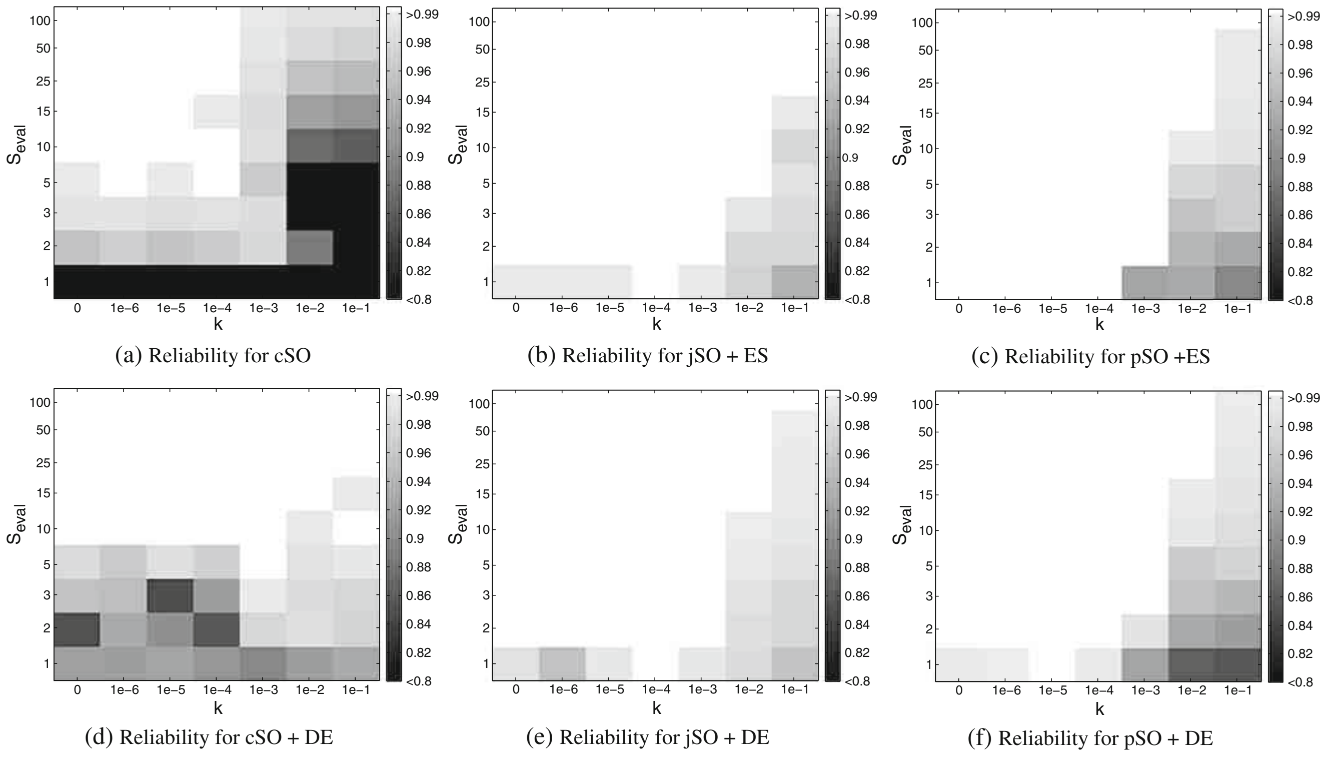

Fig. 7 Dependency of the achieved nominal reliability on S_{eval} and k for cSO, jSO, and pSO in combination with CMA-ES and DE on ${}^2h$

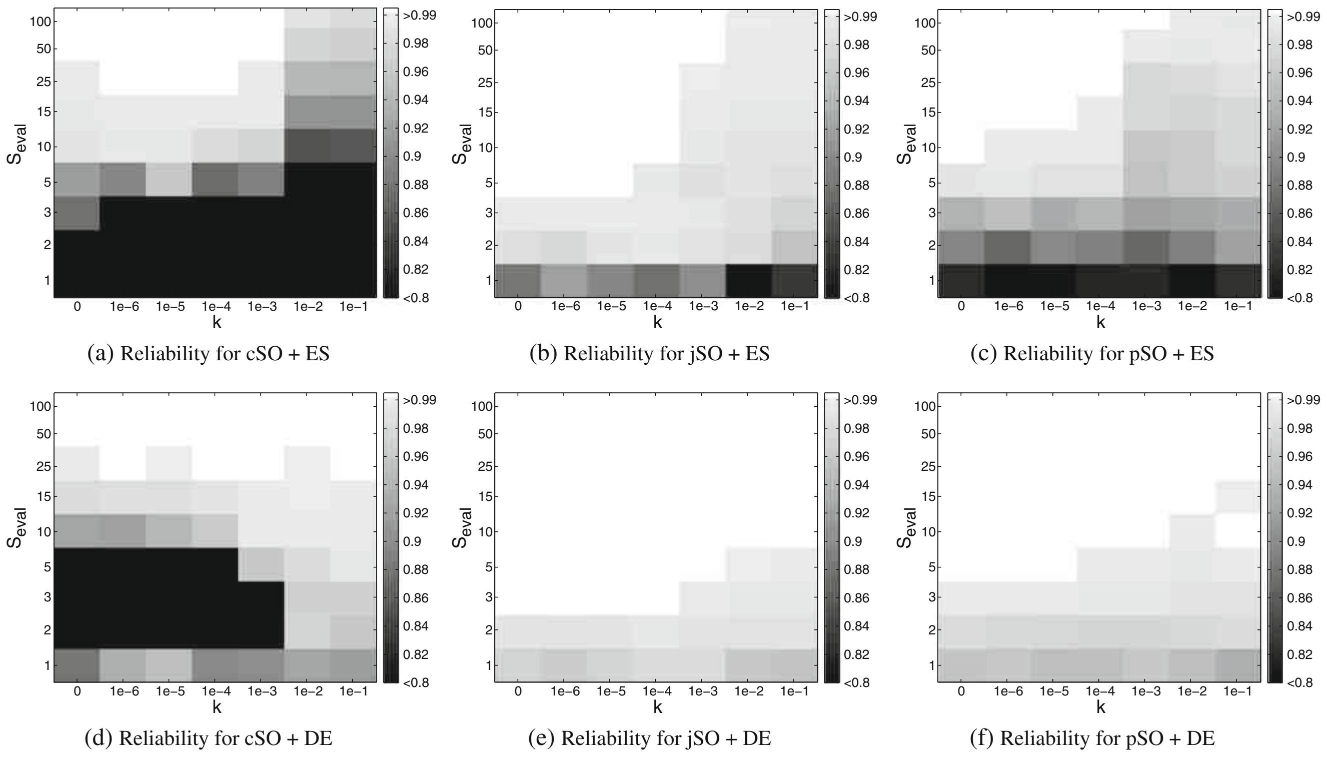

Fig. 8 Dependency of the achieved nominal reliability on S_{eval} and k for cSO, jSO, and pSO in combination with CMA-ES and DE on ${}^3h$

In the absence of information the pessimistic prior always assumes a constraint violation and results in a pessimistic overestimation of the constraint violation probability p_r. For high sample numbers the estimated p_r converges to the values obtained using the Jeffreys prior. This variant is referred to as pSO.

3.4.2 Evaluation stack size and decay factor

The evaluation stack size S_{eval} has a major impact on the reliability achieved by the classic stack ordering algorithm as shown in Bayer et al. (2010). The same has to be expected for the new proposed stack ordering approach. Since S_{eval} additionally heavily influences the achievable computational savings, a suitable compromise for S_{eval} has to be found. In order to inspect the role of this we varied S_{eval} over a wide range with

$$S_{eval} \in \{1, 2, 3, 5, 10, 15, 25, 50, 100\}.$$

By Bayer et al. (2010) the decay factor k was revealed to influence the performance of the classic stack ordering algorithm. Therefore in this study k is varied between 0 and 0.1 with:

$$k \in \{0, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1\}.$$

$k = 0$ means the absence of any decay resulting in an equal impact of all sampled constraint evaluations regardless of the number of time steps in between. On the other hand $k =$

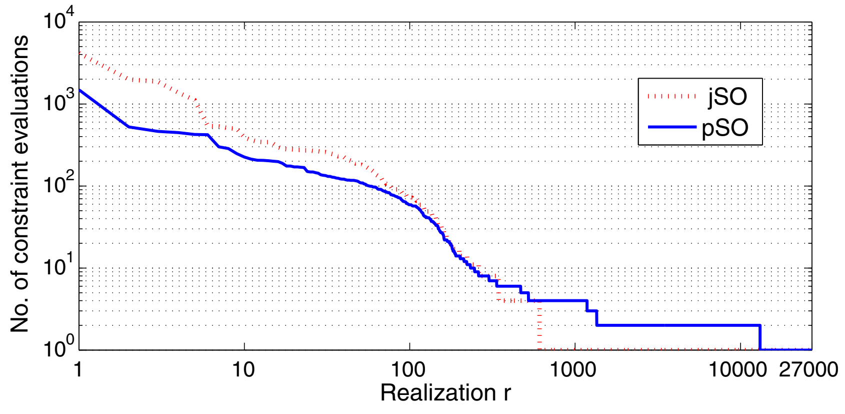

Fig. 9 Total number of actual constraint violations for each realization. The abscissae denotes the rank of the particular realization with respect to the number of actual constraint violations of the other realizations. Thereby rank 1 holds the realization with the maximum constraint violations. This evaluation was carried out on ${}^3h$ with $S_{eval} = 1$ and $k = 0$ for the new stack ordering with Jeffreys prior and pessimistic prior

Table 2 Mean and standard deviation of the achieved nominal reliabilities in % for ${}^1h$ under assumption of optimal k

SO version + optimizer		S_{eval}			
		1	2	3	5
ES	cSO	15.2 ± 3.5	96.9 ± 6.3	99.6 ± 0.4	99.9 ± 0.0
	jSO	99.9 ± 0.1	**100 ± 0**	**100 ± 0**	100 ± 0
	pSO	**100 ± 0**	**100 ± 0**	**100 ± 0**	100 ± 0
DE	cSO	88.7 ± 10.9	100 ± 0	99.9 ± 0.3	99.9 ± 0.1
	jSO	**100 ± 0**	100 ± 0	100 ± 0	100 ± 0
	pSO	99.9 ± 0.4	100 ± 0	100 ± 0	100 ± 0

0.1 yields an extreme decay with only very recent sample points having meaningful impact.

3.4.3 Parameters for the well capture problem

For the computationally demanding well capture problem, only a few stack ordering variants were selected. The new stack ordering procedure has been used with the Jeffreys prior (jSO) and the decay has been set to $k = 0$. This is favored because of the promising findings with the benchmarks. For cSO we have set the decay to $k = 0.1$, which yielded the best performance of cSO on the well capture problem in Bayer et al. (2010). To evaluate the dependency of the nominal reliability on the evaluation stack size, S_{eval} has been varied with $S_{eval} \in 2, 5, 10, 25, 50$.

3.5 Optimization algorithms

We combine all three stack ordering variants (classic SO (cSO), nSO with Jeffreys prior (jSO), nSO with pessimistic prior (pSO)) with two popular single-objective, evolutionary optimization algorithms:

1. evolution strategy with covariance matrix adaptation (CMA-ES) (Hansen and Ostermeier 2001) and
2. differential evolution (DE) (Price et al. 2005).

Table 3 Mean and standard deviation of the achieved nominal reliabilities in % for ${}^2h$ under assumption of optimal k

SO version + optimizer		S_{eval}			
		1	2	3	5
ES	cSO	24.5 ± 8.5	97.7 ± 5.8	99.2 ± 0.9	99.9 ± 0.1
	jSO	99.9 ± 0.1	**100 ± 0**	**100 ± 0**	100 ± 0
	pSO	**100 ± 0**	**100 ± 0**	**100 ± 0**	100 ± 0
DE	cSO	92.9 ± 8.6	98.6 ± 2.0	99.9 ± 0.1	100 ± 0
	jSO	**100 ± 0**	**100 ± 0**	**100 ± 0**	100 ± 0
	pSO	**100 ± 0**	**100 ± 0**	**100 ± 0**	100 ± 0

as implemented in the EvA2 optimization framework (Kronfeld et al. 2010). Both optimization algorithms have been used for a wide range of engineering problems like optimal pressure vessel design (Vu 2010), irrigation optimization (de Paly et al. 2010), and optimal well placement (Bayer et al. 2010).

For CMA-ES a comma-strategy with $\mu = 5$, $\lambda = 20$, and weighted intermediate crossover is employed. DE is applied in the DE2 variant with $F = 0.8$, $CR = 0.6$, and $\lambda = 0.6$.

Especially when used with small stack sizes, the stack ordering procedure results in a noisy constraint function. Due to the penalty function approach for constraint handling used in this study this leads to a noisy target function. Furthermore the development of the stack during the optimization adds dynamic properties to usually static optimization problems. Candidate solutions which were considered feasible by the stack ordering algorithm during the early stages of the optimization process may be infeasible later on when the stack more closely resembles the set of critical realizations. While CMA-ES is able to handle moderate noise and dynamic problems because of its non-elitist comma strategy, DE usually performs very poorly when used on noisy target functions (Krink et al. 2004). To counter this problem we combine aging and reevaluation of individuals. Individuals which survive longer in the population than a given maximum number of generations A_{max}, are assigned a new objective function value by reevaluating the constraint function based on the current stack. Lowering A_{max} results in a more dynamic behavior of DE enabling it to handle noise and dynamic effects but leads to a higher number of function evaluations. In this study we set $A_{max} = 2$ for all experiments.

Similar to the evaluation procedure for theoretical test functions we have combined stack ordering with CMA-ES and DE as optimization algorithms. Equivalent to the settings used in Bayer et al. (2010) CMA-ES was run in the default configuration as described in Hansen and Ostermeier (2001) and DE in the DE2 configuration with population size $n = 10$, $F = 0.8$, $CR = 0.6$, and $\lambda = 0.6$.

4 Results and discussion

4.1 Evaluation procedure

For each combination of stack ordering variant, k, S_{eval}, and optimization algorithm we performed 20 randomly initialized optimization runs. This includes the initial search points in the first population of the optimization algorithms, as well as the realizations in the stack. Because of the dynamic properties of the evaluated stack, which maturates during the optimization, we only use candidate solutions of the final population for the evaluation of the performance

Table 4 Mean and standard deviation of the achieved nominal reliabilities in % for ${}^3h$ under assumption of optimal k

SO version + optimizer		S_{eval}							
		1	2	3	5	10	15	25	50
ES	cSO	23.5 ±17.3	68.0±18.4	87.5 ±13.9	96.0± 7.8	99.5 ± 0.5	99.8 ± 0.3	99.9± 0	100 ± 0
	jSO	**95.2 ± 6.3**	**99.7 ± 0.4**	**99.9 ± 0.2**	**100 ± 0**	**100 ± 0**	**100 ± 0**	**100 ± 0**	100 ± 0
	pSO	92.6 ± 0.5	96.0 ± 4.1	97.8 ± 1.9	99.7 ± 0.3	99.9 ± 0.1	**100 ± 0**	**100 ± 0**	100 ± 0
DE	cSO	95.0 ± 4.1	97.0 ± 2.7	96.9 ± 2.0	99.2 ± 1.2	99.7±0.5	99.9±0.4	100.0 ± 0.1	100.0± 0
	jSO	**98.1 ± 2.4**	**99.4 ± 0.8**	**100 ± 0**	**100 ± 0**	**100 ± 0**	**100 ± 0**	100 ± 0	100 ± 0
	pSO	95.8 ± 4.4	98.2 ± 1.6	99.8 ± 0.2	**100 ± 0**	**100 ± 0**	**100 ± 0**	100 ± 0	100 ± 0

of a given optimization run. For all candidate solutions of the final population which were identified feasible (i.e. not constraint violating) by the stack ordering algorithm, we estimated the nominal reliability by evaluating the constraint function with the full set of realizations, i.e. ${}^1h$: $n_V = 1000, {}^2h : n_V = 900, {}^3h : n_V = 27000$.

The resulting mean nominal reliabilities for the different SO variants are presented in Figs. 6 for ${}^1h$, 7 for ${}^2h$, and 8 for ${}^3h$, which are organized as follows: Each Figure consists of six subfigures in two rows, with the upper row showing the results for CMA-ES and the lower one showing the results for DE. The results for the various stack ordering approaches are arranged column wise with the results for the classic stack ordering on the left. The new stack ordering when used with Jeffreys prior is reported in the middle. The results for new stack ordering with the pessimistic prior are shown on the right. Each subfigure shows a matrix of mean nominal reliabilities obtained with shrinking evaluation stack sizes S_{eval} from top to bottom and rising decays k from left to right. The actual value of the achieved nominal reliability is coded in gray scale beginning with white areas marking high nominal reliabilities (> 99 %). Subsequently lower nominal reliabilities are marked by gradually darker areas. Black areas indicate unsatisfactory nominal reliabilities below 80 %.

4.2 General performance of stack ordering

Comparing the subfigures (a), (b), (c) respectively (d), (e), (f) for each of the Figs. 6, 7, and 8 shows that the new generalized versions, jSO and pSO, outperform classic cSO for all inspected combinations of benchmark problem and optimization algorithm. Variant cSO requires significantly larger stack sizes S_{eval} than the two proposed new stack ordering variants in order to achieve comparable nominal reliabilities. This indicates that the approximation of the constraint violation probabilities p_r by the new stack ordering approach allows a superior determination of the actual set of critical realizations compared to the heuristic credit assignment approach of cSO. In essence, the new approach enables to determine the feasibility of a given solution candidate with lower amount of function evaluations.

The difference between jSO and pSO is much less significant with a slight advantage of jSO in most of the evaluated cases. Especially in the case of the challenging benchmark constraint ${}^3h$ the more precise approximation of p_r by jSO results in a better representation of the critical set in the stack than the pessimistic approach of pSO. The approximation based on Jeffreys prior by jSO concentrates the sequence constraint evaluations on a smaller set of realizations compared to the pessimistic prior. As

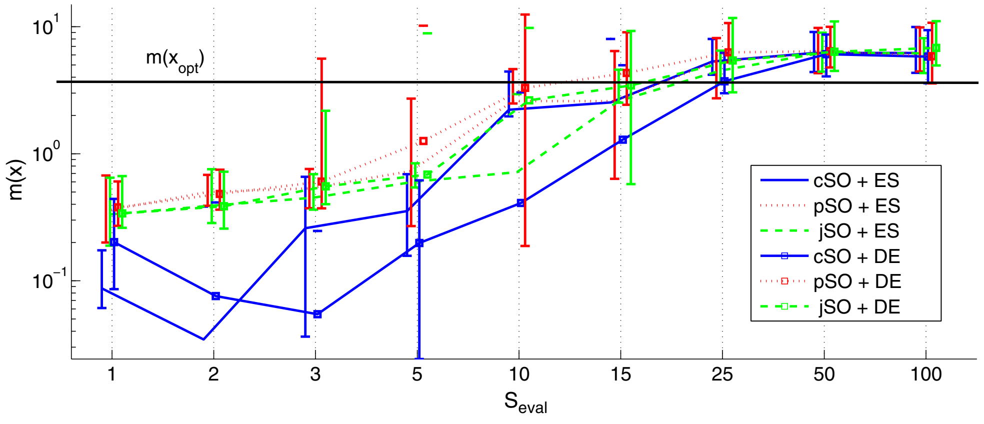

Fig. 10 Dependency of the objective function value on S_{eval} for cSO, jSO and pSO on ${}^3h$

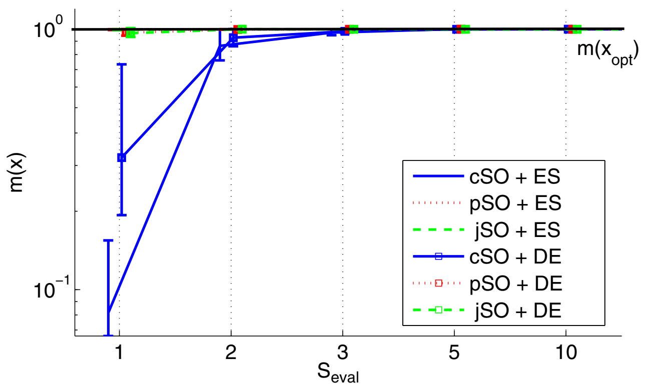

Fig. 11 Dependency of the objective function value on S_{eval} for cSO, jSO and pSO on ${}^1h$

depicted by Fig. 9, the 100 most evaluated realizations have been evaluated up to 5 times more often by jSO than by pSO. In contrast pSO yields a broader distribution of constraint evaluations over all realizations. This means pSO acts exploratory while jSO is more exploitation focused. In theory, the exploratory behavior of pSO should facilitate an easier detection of changes within the set of critical realizations if the search region of interest, which is predominantly sampled by the optimization algorithm, moves during the optimization process. For the benchmark constraints of our study, however, better results are generated by the more focused constraint evaluation provided by jSO. This may be caused by the simple hypersphere function used in the study that results in a rapid convergence of the search.

4.3 Dependency on evaluation stack size

Tables 2, 3, and 4 give a more detailed insight into the dependency of the derived nominal reliability on the specified size of the evaluation stack S_{eval}. Each entry provides

Table 5 Mean number of constraint function evaluations with minimum required S_{eval} (stated in brackets) in order to achieve maximum nominal reliability for each SO variant and benchmark constraint with CMA-ES

Benchmark	cSO	jSO	pSO	Full evaluation
${}^1h$	$7.2 \cdot 10^4$ (10)	$1.7 \cdot 10^4$ (2)	$1.2 \cdot 10^4$ (2)	10^7
${}^2h$	$6.5 \cdot 10^4$ (10)	$1.7 \cdot 10^4$ (2)	$1.2 \cdot 10^4$ (2)	$9 \cdot 10^6$
${}^3h$	$5.0 \cdot 10^5$ (50)	$6.4 \cdot 10^4$ (5)	$1.7 \cdot 10^5$ (15)	$2.7 \cdot 10^8$

the mean nominal reliability as well as its standard deviation under the assumption of an optimal k (refer to Section 4.4 for details on optimality) for a given combination of S_{eval}, stack ordering variant, and optimization algorithm. For each S_{eval}, and optimization algorithm the best entries as well as entries which do not differ significantly from the best one (p-value > 0.05) are highlighted in bold letters.

In almost all cases larger stack sizes yield an improved nominal reliability until maximum achievable nominal reliability is reached. While jSO and pSO arrive at this point with $S_{eval} = 2$ or even $S_{eval} = 1$ on ${}^1h$ and ${}^2h$, cSO requires larger stack sizes even for these deliberately chosen simple problems. As a consequence of its complexity ${}^3h$ requires stack sizes that are considerably larger for all stack ordering variants. Nevertheless the minimum required stack size for jSO and pSO of $S_{eval} = 5$ when combined with DE and $S_{eval} = 5$ for jSO and $S_{eval} = 15$ for pSO when combined with CMA-ES require significantly less constraint evaluations than cSO with a minimum required stack size of $S_{eval} = 25$ (DE) respectively $S_{eval} = 50$ (CMA-ES).

Because of the reduced noise caused by larger evaluation stack sizes, S_{eval} also has an impact on the best objective function values ($m(\mathbf{x})$) achieved as shown by the Figs. 10(${}^3h$), 11 (${}^1h$), and 12 (${}^2h$). The figures show the mean of the best objective function values $m(\mathbf{x})$ for the

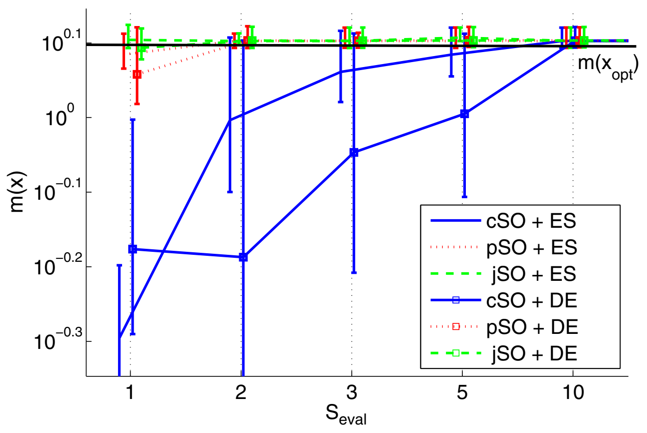

Fig. 12 Dependency of the objective function value on S_{eval} for cSO, jSO and pSO on ${}^2h$

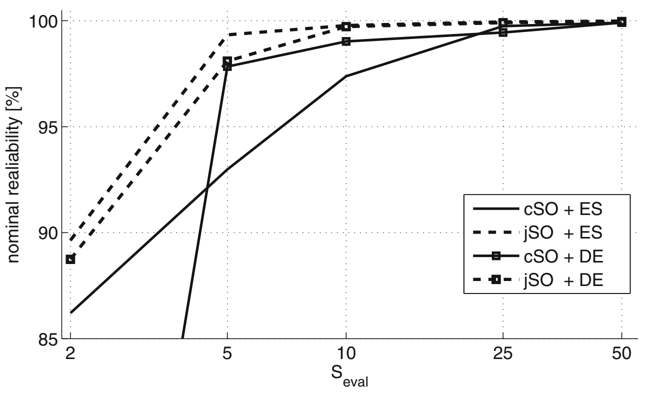

Fig. 13 Dependency of the nominal reliability on S_{eval} for the well capture problem after 500 objective function evaluations

Table 6 Mean and standard deviation of the achieved nominal reliabilities for the well capture problem after 500 objective function evaluations

SO version + optimizer		S_{eval}				
		2	5	10	25	50
ES	cSO	86.61 ± 11.02	92.98 ± 8.66	97.38 ± 5.82	99.74 ± 0.32	99.90 ± 0.16
	jSO	89.63 ± 11.02	99.33 ± 0.83	99.78± 0.26	99.94 ± 0.08	99.98 ± 0.04
DE	cSO	52.99 ± 19.73	97.83 ± 2.10	99.02 ± 1.48	99.44 ± 2.01	99.91 ± 0.12
	jSO	88.476 ± 11.9	98.09 ± 3.00	99.72± 0.33	99.90 ± 0.12	99.96 ± 0.08

examined combinations of optimization algorithm, stack ordering method, and evaluation stack size. The 10th and 90th percentiles are denoted by vertical bars. The objective function value of the theoretical optimal non violating solution x^{opt} for a given constraint (refer to section 3.2) is shown as black horizonal line. For ^1h and ^2h the impact on the fitness stems from a compromised reliability caused by small stack sizes. Due to the excellent performance of jSO and pSO on ^1h and ^2h this effect is only visible for cSO.

As depicted in Fig. 10 this effect also dominates on ^3h for all SO variants with evaluation stack sizes smaller than the minimum required stack size to achieve maximum reliability. But unlike ^1h and ^2h, evaluation stack sizes above the minimum required stack size further negatively impact the achieved objective function value for ^3h. This is caused by the disjoint search space which requires traversing infeasible regions in order to reach the feasible global optimizers (see Fig. 5). This can be more easily done by the overlying search algorithm when actualy infeasible candidate solutions are misclassified as feasible by the stack ordering algorithm. Because of this misclassification low values for S_{eval} make the 100 % reliability region appear bigger than it actually is and potentially result in a virtual connection of the disjoint feasible regions of the search space.

Without specialized constraint optimization techniques the small feasible regions surrounding the global optimizers are almost unreachable by the optimization algorithms employed in this study when combined with SO using large S_{eval}.

At first glance cSO when combined with DE does not follow the general relationship of higher nominal reliability with increasing S_{eval}. Up to a certain threshold cSO+DE with low decay achieves a better nominal reliability with $S_{eval} = 1$ than with $S_{eval} \geq 2$. The reason for this is the credit assignment rule of cSO which does not assign any credit to the first realization in the evaluation stack (see (9)). Therefore for $S_{eval} = 1$ all realization are equally likely to be chosen for constraint evaluation and only one realization is evaluated, cSO operates like an algorithm that randomly selected a single realization in order to determine the feasibility of a candidate solution. This yields an extremely noisy constraint function. In contrast, a larger, yet too small evaluation stack combined with small decay rates k results in a completely static behavior of cSO. As soon as S_{eval} realizations have accumulated a credit of at least $C_r > \ln(S_{eval})$ only these realizations are evaluated for each candidate solution without any chance of other realizations entering the stack (see (7)). For small S_{eval} and small k this may happen very early in the optimization process. When combined with DE for cSO the negative effect of an early stagnation of the stack seems to outweigth the impact of the extremely noisy constraint function resulting from $S_{eval} = 1$.

4.4 Dependency on decay

On the one hand applying a small decay should allow stack ordering to adapt the estimated constraint violation probabilities during the optimization process onto a moving distribution of candidate solutions as it puts less emphasis on old constraint evaluations. On the other hand, because of the resulting loss of information, decaying n_r and c_r may reduce the quality of the estimated constraint violation probabilities. Figures 6 and 7 show that for jSO and pSO there may be a beneficial small decay of $k = 10^{-4}$ when used

Table 7 Mean number of model evaluations for the well capture problem after 500 objective function evaluations

SO version+ optimizer		S_{eval}				
		2	5	10	25	50
ES	cSO	799 ± 26	1674 ± 120	3098 ± 327	7558 ± 703	16020 ± 1015
	jSO	811 ± 25	1739 ± 136	3205± 260	8017 ± 516	16051 ± 734
DE	cSO	875 ± 35	1676 ± 182	2875 ± 389	6965 ± 713	14560 ± 1808
	jSO	761 ± 25	1481 ± 93	2754± 388	7272 ± 863	14669 ± 1095

Table 8 Mean number of objective function evaluations for the well capture problem after 10,000 model evaluations

SO version+ optimizer		S_{eval}				
		2	5	10	25	50
ES	cSO	6232 ± 351	3101 ± 203	1642 ± 125	676 ± 69	303 ± 20
	jSO	5636 ± 438	2753 ± 656	1581± 258	619 ± 30	311 ± 21
DE	cSO	8280 ± 788	6204 ± 1917	4072± 2312	767 ± 151	344 ± 42
	jSO	7501 ± 1058	4625 ± 2092	2005 ± 1667	682 ± 56	347 ± 24

with the minimum stack size of $S_{eval} = 1$. Decay values above this threshold have a negative impact on the achieved nominal reliabilities. On the more complex benchmark constraint ${}^3h$ there is no positive impact of any decay on the performance of jSO and pSO. Regardless of the evaluation stack size the highest nominal reliabilities have been achieved with $k = 0$. These results indicate that it may be advantageous to completely omit the decay factor when employing jSO and pSO. This allows to reduce the number of parameters that have to be set by the user to one—the stack size, S_{eval}.

On cSO the impact of decay is less clear cut and depends on benchmark function and optimization algorithm. When combined with CMA-ES, cSO always performed best without any decay whereas DE benefitted greatly from comparatively high decay values $k \geq 10^{-2}$ on ${}^2h$ and ${}^3h$ as shown in Figs. 7d and 8d.

4.5 Computational savings

The number of model evaluations is highly correlated with the size of the evaluation stack S_{eval}. Therefore S_{eval} should be kept as low as possible while still achieving maximum nominal reliability R_{max}. This means, in order to achieve the required maximum nominal reliability no realization must be constraint violating for a given candidate solution. Table 5 gives the mean number of function evaluations required by cSO, jSO, and pSO in order to achieve R_{max} in each of the 20 trial runs for each of the three benchmark constraints. For comparison we also provide the number of model evaluations that is required if all realization would be evaluated for each candidate solution during optimization. jSO and pSO perform very similar on ${}^1h$ and ${}^2h$ and on average require 4 to 6 times less model evaluations than cSO. On ${}^3h$ the difference between jSO and cSO becomes even more pronounced with the later requiring up to 8 times as many model evaluations in order to achieve R_{max}. pSO is still 3 times more efficient than cSO. When compared against a full evaluation of all realization jSO and pSO reduced the number model evaluations by a factor of 500 to 800 on ${}^1h$ and ${}^2h$ and an even larger factor of up to 4300 on ${}^3h$ in the case of jSO. This result underlines the potential computational savings that can be achieved by determining the set of critical realization especially for use cases with large realization sets.

4.6 Well capture problem

4.6.1 Evaluation procedure

Each of the 20 possible combinations for the chosen stack ordering procedures, evolutionary algorithms and stack sizes was run 20 times with random initialization resulting in 400 overall optimization runs. A fixed set of 1000 realizations was generated and not changed for the initial stack of each optimization run. The optimization runs were stopped after 500 objective function evaluations. In conjunction with large S_{eval} this proved to be sufficient to achieve high nominal reliabilities at the well capture problem as described in Bayer et al. (2010). For performance assessment and comparison of the different algorithms, the nominal reliability of the best individuals of the 5 final generations was determined. The overall computational time was roughly 100 days, because of the high computational demand of the hydrogeological model. Particular time consuming was the performance assessment of the optimization results, for which the conditions for all 1000 realizations of hydraulic conductivity had to be modeled repeatedly.

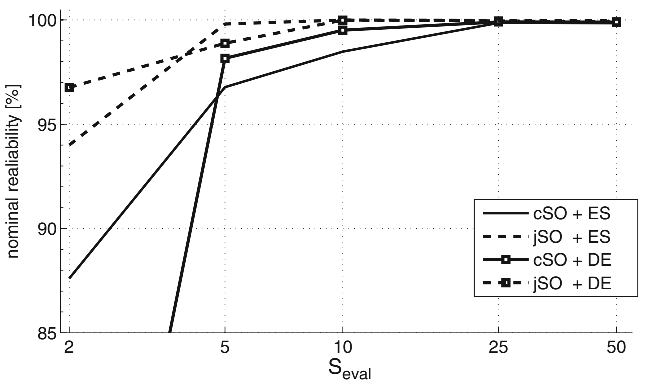

Fig. 14 Dependency of the nominal reliability on S_{eval} for the well capture problem after 10,000 model evaluations

Table 9 Mean and standard deviation of the achieved nominal reliabilities for the well capture problem after 10,000 model evaluations

SO version+ optimizer		S_{eval}				
		2	5	10	25	50
ES	cSO	87.61 ± 14.02	96.78 ± 3.81	98.48 ± 2.70	99.86 ± 0.26	99.85 ± 0.21
	jSO	94.00 ± 15.02	99.81 ± 0.31	99.99± 0.02	99.98 ± 0.04	99.96 ± 0.12
DE	cSO	61.09 ± 16.55	98.16 ± 2.29	99.51 ± 0.61	99.90 ± 0.11	99.89 ± 0.13
	jSO	96.76 ± 7.78	98.88 ± 3.94	99.99± 0.03	99.96 ± 0.10	99.90 ± 0.14

4.6.2 Results

Figure 13 shows the mean of the nominal reliabilities achieved by the evaluated combinations of stack ordering procedure and evolutionary algorithms for the various stack sizes. Table 6 provides a more detailed picture and also states the measured standard deviations.

The results on the well capture problem largely mirror the findings from the theoretical test cases. Except for high S_{eval}, where both SO variants achieve almost 100 % reliability, jSO outperforms cSO for each of the considered S_{eval} by a considerable margin regardless of the overlying evolutionary algorithm. For jSO an evalution stack size of $S_{eval} = 10$ is sufficient to achieve almost perfect nominal reliability while cSO requires a S_{eval} of 25 or 50 depending on the overlying optimization algorithm. One exception is the performance of cSO in conjunction with DE for $S_{eval} = 5$ which achieves roughly the same nominal reliability as its jSO counterpart.

An interesting aspect is the differing performance of the overlying evolutionary algorithms. While ES consistently outperforms DE in conjunction with jSO, DE shows better performance than CMA-ES for cSO with $S_{eval} = \{5, 10\}$. This may be attributed to the higher susceptibility of CMA-ES to noise than DE in combination with aging and reevaluation. Especially for low values of S_{eval} cSO produces considerably higher noise than jSO resulting in degraded performance of CMA-ES.

For the same number of objective function evaluations the stack ordering procedure results in a lower number of model evaluations for smaller stack sizes as shown in Table 7. To ensure that the better performance of larger stack sizes is not solely caused by this effect we also evaluated the performance of all optimization runs after 10,000 model evaluations, which results in considerably larger number of objective function evaluations for cases with small S_{eval} as shown in Table 8. Figure 14 and Table 9 show the resulting mean nominal reliabilities for all considered combinations of stack order procedure, evolutionary algorithm, and stack size.

While the nominal reliability improved for cases with $S_{eval} = 2$ the overall picture is unchanged and almost perfectly resembles the results after 500 objective function evaluations. This shows that the new stack ordering procedure does indeed result in considerable computational savings over the classic stack ordering procedure.

5 Conclusion

The classic version of stack ordering has previously been developed for the specific needs of combined simulation-optimization in the area of hydrogeology. The heuristic multi-model sampling technique to enable fast detection of highly reliable solutions by approximate objective function evaluations revealed to be suitable for such problems when it is coupled with a single-objective optimizer, typically evolutionary algorithm. Stack ordering dynamically determines during optimization which subset of simulation model variants to choose for estimating constraint evaluation. By limiting evaluations to a subset (the so-called evaluation stack) of cardinality 10 to 50 rather than evaluating the full set considerable computational savings can be achieved. Still, the classic version has not been tested on general benchmarks, and it is purely empirical based. In the presented work, a new version of stack ordering is introduced that represents an improved sampling algorithm based on Bayesian principles. We defined three benchmarks of increasing complexity of the given stochastic constraints. The latter are described by a finite and large set (i.e. the “stack”) of constraint realizations. The objective function has to be minimized while constraint violation has to be avoided, simulating high-reliability based design or risk averse combined simulation-optimization problems. The new stack ordering samples the stack according to deterministic rules. Realizations are selected for the current evaluation stack with highest expected probability of constraint violation, assuming that this can be described by a Bernoulli process. Two different variants, which differ with respect to the prior, are compared on the benchmarks to the classic version. Results on theoretical benchmarks and the real-world oriented well optimization problem show that computational savings can be increased with the new stack ordering by a factor of about 5 when compared to the classic stack ordering variant.

For the examined benchmark problems the new algorithm worked efficiently with decay being generally set to zero. The new stack ordering has only one free parameter (the evaluation stack size S_{eval}) that has to be set by the user.

Even if the presented findings are encouraging, deeper analysis on performance and application window of the presented algorithm is necessary. For this, more complex benchmarks need to be considered, and also additional real-world applications have to be tested. Further work is planned to modify the new stack ordering approach to also support reliability based optimization with less than 100 % reliability. In engineering optimal solutions at low quantiles are often preferred to potentially overdesigned solutions and if safety requirements are more moderate. The potential of stack ordering has to be revealed as well by comparison to alternative concepts of high-reliability based design, although such alternatives are often formulated for a specific case or practical problem. So far, general approaches for tackling computationally demanding multi model or multi realization based optimization problems are rare.

Acknowledgments We thank the two reviewers who helped to improve the paper and Gabi Moser for proofreading.

References

Arnst M, Ghanem R, Soize C (2010) Identification of bayesian posteriors for coefficients of chaos expansions. J Comput Phys 229(9):3134–3154. doi:10.1016/j.jcp.2009.12.033. http://www.sciencedirect.com/science/article/B6WHY-4Y34SWD-2/2/f6537e3705bb6566ef2e6717b0f8f7b5

Avigad G, Branke J (2008) Embedded evolutionary multi-objective optimization for worst case robustness. In: Ryan C, Keijzer M (eds) GECCO, ACM, pp. 617–624

Avigad G, Coello CAC (2010) Highly reliable optimal solutions to multi-objective problems and their evolution by means of worst-case analysis. Eng Optim 42(12):1095–1117. http://www.informaworld.com/10.1080/03052151003668151

Azadivar F (1999) Simulation optimization methodologies. In: Proceedings of the 31st conference on winter simulation: simulation—a bridge to the future, vol 1 ACM, New York, WSC '99, pp 93–100. http://doi.acm.org/10.1145/324138.324168

Bayer P, Bürger C, Finkel M (2004) Evolutionary algorithms for the optimization of advective control of contaminated aquifer zones. Water Resour Res 40(1146):W06506. doi:10.1029/2003WR002675

Bayer P, Bürger CM, Finkel M (2008) Computationally efficient stochastic optimization using multiple realizations. Adv Water Resour 31(2):399–417. doi:10.1016/j.advwatres.2007.09.004. http://www.sciencedirect.com/science/article/B6VCF-4PT0Y85-1/2/0b7d5cdbd9868f3a5127c7d306968808

Bayer P, de Paly M, Bürger CN (2010) Optimization of high-reliability-based hydrological design problems by robust automatic sampling of critical model realizations. Water Resour Res 46(5):W05504. doi:10.1029/2009WR008081

Binder K (1979) Monte Carlo methods in statistical physics / with contributions by K. Binder... [et al.]. In: Binder K (ed). Springer-Verlag, Berlin, New York

Cirpka OA, Bürger CM, Nowak W, Finkel M (2004) Uncertainty and data worth analysis for the hydraulic design of funnel-and-gate systems in heterogeneous aquifers. Water Resour Res 40(11):W11502. doi:10.1029/2004WR003352

Debusschere BJ, Najm HN, Pébay PP, Knio OM, Ghanem RG, Maître OPL (2005) Numerical challenges in the use of polynomial chaos representations for stochastic processes. SIAM J Sci Comput 26:698–719. doi:10.1137/S1064827503427741

de Paly M, Schuetze N, Zell A (2010) Determining crop-production functions using multi-objective evolutionary algorithms. In: Proceedings of the IEEE congress on evolutionary computation (CEC), pp 1870–1877, Barcelona, Spain. doi:10.1109/CEC.2010.5586147

Du X, Guo J, Beeram H (2008) Sequential optimization and reliability assessment for multidisciplinary systems design. Struct Multidiscip Optim 35:117–130. doi:10.1007/s00158-007-0121-7

Fu MC, Glover F, April J (2005) Simulation optimization: a review, new developments, and applications. In: Winter simulation conference. ACM, pp 83–95.

Fink D (1997) A Compendium of Conjugate Priors. Tech. rep., Montana State Unieristy, URL http://www.johndcook.com/CompendiumOfConjugatePriors.pdf

Ghanem RG, Spanos PD (1991) Stochastic finite elements: a spectral approach. Springer-Verlag, New York

Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195

Harbaugh A, Banta E, Hill M, McDonald M (2000) MODFLOW-2000, the U.S. Geological Survey modular ground water model, user guide to modularization concepts and the ground water flow process

Hasofer AM, Lind NC (1974) Exact and invariant second-moment code format. J Eng Mech Div 100(1):111–121

Hubbard WD (2007) How to measure anything finding the value of 'intangibles' in business. Wiley, Hoboken

Jeffreys H (1946) An invariant form for the prior probability in estimation problems. Proc Royal Soc Lond Ser A Math Phys Sci 186(1007):453–461. doi:10.2307/97883

Kourakos G, Mantoglou A (2008) Remediation of heterogeneous aquifers based on multiobjective optimization and adaptive determination of critical realizations. Water Resour Res 44(12):W12408. doi:10.1029/2008WR007108

Krink T, Filipic B, Fogel G, Thomsen R (2004) Noisy optimization problems - a particular challenge for differential evolution? In: Proceedings of 2004 congress on evolutionary computation, pp 332–339.IEEE Press

Kronfeld M, Planatscher H, Zell A (2010) The EvA2 optimization framework. In: Blum C, Battiti R (eds) Learning and intelligent optimization conference, special session on software for optimization (LION-SWOP). Springer Verlag, Venice, Italy. no. 6073 in lecture notes in computer science, LNCS, pp 247–250. http://www.ra.cs.uni-tuebingen.de/publikationen/2010/Kron10Ev

Lin G, Tartakovsky A (2009) An efficient, high-order probabilistic collocation method on sparse grids for three-dimensional flow and solute transport in randomly heterogeneous porous media. Adv Water Resour 32(5):712–722. doi:10.1016/j.advwatres.2008.09.003. http://www.sciencedirect.com/science/article/B6VCF-4TJ6FP4-1/2/7e4ecf6e3963323bede48fc4e9fa2fc3, dispersion in Porous Media

Lu Z, Zhang D (2005) A comparative study on uncertainty quantification for flow in randomly heterogeneous media using monte carlo simulations and conventional and kl-based moment-equation approaches. SIAM J Sci Comput 26:558–577. doi:10.1137/S1064827503426826

Lu Z, Zhang D (2007) Stochastic simulations for flow in non-stationary randomly heterogeneous porous media using a kl-based moment-equation approach. Multiscale Model Simul 6(1):228–245. doi:10.1137/060665282. http://link.aip.org/link/?MMS/6/228/1

Mantoglou A, Kourakos G (2007) Optimal groundwater remediation under uncertainty using multi-objective optimization. Water Resour Manag 21:835–847. doi:10.1007/s11269-006-9109-0

Morgan DR, Eheart JW, Valocchi AJ (1993) Aquifer remediation design under uncertainty using a new chance constrained programming technique. Water Resour Res 29(3):551–561. doi:10.1029/92WR02130

Nicklow J, Reed P, Savic D, Dessalegne T, Harrell L, Chan-Hilton A, Karamouz M, Minsker B, Ostfeld A, Singh A, on Evolutionary Computation in Environmental EZATC, Engineering WR (2010) State of the art for genetic algorithms and beyond in water resources planning and management. J Water Resour Plan Manag 136(4):412–432. doi:10.1061/(ASCE)WR.1943-5452.0000053. http://link.aip.org/link/?QWR/136/412/1

Pollock DW (1994) Users guide for modpath/mod-path-plot, version a particle tracking post-processing package for modflow, the U.S. geological survey finite-difference ground-water flow

Price KV, Storn RM, Lampinen JA (2005) Differential evolution a practical approach to global optimization. Natural computing series. Springer-Verlag, Berlin

Singh A, Minsker BS (2008) Uncertainty-based multiobjective optimization of groundwater remediation design. Water Resour Res 44(2):W02404. doi:10.1029/2005WR004436

Smalley JB, Minsker BS, Goldberg DE (2000) Risk-based in situ bioremediation design using a noisy genetic algorithm. Water Resour Res 36(10):3043–3052. doi:10.1029/2000WR900191

Tvedt L (1990) Distribution of quadratic forms in normal space - application to structural reliability. J Eng Mech 116(6):1183–1197. doi:10.1061/(ASCE)0733-9399(1990)116:6(1183)

Valdebenito MA, Schuëller GI (2010) A survey on approaches for reliability-based optimization. Struct Multidiscip Optim 42:645–663. doi:10.1007/s00158-010-0518-6

Vu V (2010) Minimum weight design for toroidal pressure vessels using differential evolution and particle swarm optimization. Struct Multidiscip Optim 42:351–369. doi:10.1007/s00158-010-0494-x

Wagner BJ, Gorelick SM (1989) Reliable aquifer remediation in the presence of spatially variable hydraulic conductivity: from data to design. Water Resour Res 25(10):2211–2225. doi:10.1029/WR025i010p02211

Wu J, Zheng C, Chien CC, Zheng L (2006) A comparative study of Monte Carlo simple genetic algorithm and noisy genetic algorithm for cost-effective sampling network design under uncertainty. Adv Water Resour 29:899–911. doi:10.1016/j.advwatres.2005.08.005

Xi Z, Hu C, Youn B (2011) A comparative study of probability estimation methods for reliability analysis. Struct Multidiscip Optim:1–20. doi:10.1007/s00158-011-0656-5

Zhang D, Lu Z (2004) An efficient, high-order perturbation approach for flow in random porous media via karhunen-lo & 232;ve and polynomial expansions. J Comput Phys 194:773–794. doi:10.1016/j.jcp.2003.09.015. http://portal.acm.org/citation.cfm?id=1008428.1008445