Available online at www.sciencedirect.com
ScienceDirect
Advances in Water Resources xxx (2007) xxx–xxx
Advances in Water Resources
www.elsevier.com/locate/advwatres

Computationally efficient stochastic optimization using multiple realizations

P. Bayer [*], C.M. Bürger, M. Finkel

Center for Applied Geosciences, University of Tübingen, Sigwartstrasse 10, D-72076 Tübingen, Germany

Received 21 May 2007; received in revised form 25 September 2007; accepted 25 September 2007

Abstract

The presented study is concerned with computationally efficient methods for solving stochastic optimization problems involving multiple equally probable realizations of uncertain parameters. A new and straightforward technique is introduced that is based on dynamically ordering the stack of realizations during the search procedure. The rationale is that a small number of critical realizations govern the output of a reliability-based objective function. By utilizing a problem, which is typical to designing a water supply well field, several variants of this "stack ordering" approach are tested. The results are statistically assessed, in terms of optimality and nominal reliability. This study demonstrates that the simple ordering of a given number of 500 realizations while applying an evolutionary search algorithm can save about half of the model runs without compromising the optimization procedure. More advanced variants of stack ordering can, if properly configured, save up to more than 97% of the computational effort that would be required if the entire number of realizations were considered. The findings herein are promising for similar problems of water management and reliability-based design in general, and particularly for non-convex problems that require heuristic search techniques.

Keywords: Stochastic optimization; Reliability-based design; Wellhead protection; Evolutionary algorithms; CMA-ES

1. Introduction

The use of multiple, equally probable realizations of an uncertain parameter set, instead of relying exclusively on one deterministic characterization, is a common method of accounting for the incomplete knowledge in model input parameter values (e.g., [11]). If flow or transport models are applied in the decision making process for groundwater management, proactive and post-mortem procedures involving multiple realizations can be distinguished (e.g., [38]). Solving a management problem in a proactive way means that probability distributions are processed in the objective function (OF). Here, the OF to be optimized represents a combination of the results simultaneously obtained for numerous realizations. Post-mortem techniques, such as sensitivity analyses, examine one or more solutions found by applying a deterministic model first (e.g., [36,6]). Then, the susceptibility of these solutions to model inaccuracies is examined by assuming statistically distributed parameters, for example by Monte Carlo analysis. In this way, highly influential model parameters are identified.

In this paper, we will focus on the proactive concept of reliability-based optimization (RBO) as it is suited to condition solutions a priori for being insusceptible to parameter uncertainty. The purpose of RBO is the identification of one design which is suited to all realizations simultaneously. For example, specific OF values are computed for each realization and then their mean or the worst value is optimized. Formally, one OF evaluation requires sequential/parallel runs of the model for the entire stack of

[*] Corresponding author.
E-mail addresses: peter.bayer@uni-tuebingen.de (P. Bayer), claudius.buerger@uni-tuebingen.de (C.M. Bürger), michael.finkel@uni-tuebingen.de (M. Finkel).

0309-1708/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advwatres.2007.09.004

NOTATION

A	penalty variable for drawdown
B	penalty variable for groundwater residence time
C^*	criterion for determination of sampling probability
CMA-ES	evolution strategy with covariance matrix adaptation
C_r	credit of realization r
$d_{\mathrm{act},r}$	drawdown violation at extraction well locations of a realization (m)
$d_{\max}$	maximum drawdown violation (m)
d_r	relative drawdown violation for realization r
f	iteration threshold for SORed
F	number of realizations indicating failure
GA	genetic algorithm
i	row coordinate of well, within ranges of $i_{\min} < i < i_{\max}$
j	column coordinate of well, within ranges of $j_{\min} < j < j_{\max}$
k	decay factor
n	problem dimension
NO	no ordering of realizations
NORep	no ordering of realizations with replacement
OF	objective function, OF_r is objective function evaluated for realization r
OR	optimization run
pen	penalty
p_r	sampling probability for realizations in SORed, SORep and SORepDecay
q	extraction rate (l/s), within ranges of $q_{\min} < q < q_{\max}$
r	realization index
RBO	reliability-based optimization
R_{N}	nominal reliability
$\langle R \rangle$	true reliability
S	stack size
S_{eval}	size of evaluation stack
SO	stack ordering (and break)
SORed	stack ordering and reduction
SORep	stack ordering with replacement
SORepDecay	stack ordering with replacement and decay
s_r	position of realization r in sorted stack
t	iterations of optimization algorithm (successive OF evaluations), $t = 1, \ldots, t_{\max}$
$z_{\mathrm{act},r}$	residence time violation at wells of a realization (d)
$z_{\min}$	minimum residence time violation (d)
z_r	relative residence time violation for realization r
Ω	design constraint for groundwater management problem
λ	size of population (CMA-ES)
μ	size of offspring (CMA-ES)
σ_{Y}^2	variance of log hydraulic conductivity

realizations. Reliable solutions are those which comply with the constraints of all realizations. In contrast, robust designs are those that are least sensitive to the expected uncertainty of input parameters. Despite the apparent computational intensity of a multiple realizations based optimization procedure, it is appealing due to its potential to deliver robust or reliable solutions, its straightforwardness and flexibility. The potential of this approach has been demonstrated in various applications for RBO in the field of groundwater management [47,43,37,17,3], as well as in other disciplines related to reservoir management, reliability-based engineering and structural safety [16,42].

This study proposes computationally efficient methods of achieving optimal solutions for a design problem involving multiple realizations. The purpose is to maximize both solution reliability and search efficiency under the assumption that a given stack of realizations exists that adequately represents the possible space of parameter uncertainty or imprecision. We will present a new procedure, ‘‘stack ordering’’, which is based on dynamically ordering realizations during the optimization process. Several variants of the method will be introduced and tested for their capability to optimally configure a water supply well field under uncertain heterogeneous aquifer conditions.

2. Stochastic optimization in groundwater management: related work

2.1. Stacking method

A major source of uncertainty in groundwater management is the spatial distribution of hydraulic conductivity. Commonly, only a small fraction of an aquifer is sampled. If point measurements are available, calibrating a model means interpolating between these points of known parameters by estimating the missing unknown hydraulic conductivity of the sampled features. The various possibilities of interpolation can be reflected by multiple, equally probable realizations of the hydraulic conductivity distributions stacking method, [47]. An optimal control solution is then assessed according to its performance under various probable hydraulic conditions, which are represented by the realizations.

If hydraulic conductivity is sampled from its distribution curve, an infinite number of realizations are necessary in order to obtain an exact representation of the probability space. Therefore, multiple-realization based optimization demands a compromise between desired accuracy and computational feasibility. At the same time, even if it were possible to reach maximum accuracy in describing parameter uncertainty, there are numerous other factors encroaching upon the preciseness of model predictions. A main factor here is that groundwater models will always be abstract representations of reality with specific conceptual shortcomings. A second important factor is an inherent degree of imprecision in the measurements. Compared with parameter uncertainty, conceptual and measurement imprecision appears to be extremely hard to quantitatively describe. This is reflected in the relatively few studies dedicated to those issues other than the consequence of model parameter uncertainty (e.g., [35,25,29]).

In view of the variable sources of model inaccuracies, the multiple realization concept is appealing as several different factors causing model inaccuracy may be considered in a straightforward manner. This may be seen as one major advantage over approximate analytic approaches, which merely address parameter uncertainties such as first-order or second-order reliability methods (e.g., [30,14,27,2]).

Although the resolution of expected parameter spaces (and their representation of reality) is limited by the computational feasibility, a satisfactory exactness of model outputs may already be achieved by a small stack of realizations. This was shown by Chan [13] who derived two statistical models based on Bayesian analysis as well as on order statistics to estimate the reliability of hydraulic management problem solutions. If S is the number of realizations of multiple random hydraulic conductivity fields, i.e., the stack size, the expected true reliability $\langle R \rangle$ of a solution which satisfies the constraints of all realizations is $S/(S+1)$ or $(S+1)/(S+2)$, respectively. Based on a comprehensive empirical analysis, Feyen and Gorelick [17] revealed deficiencies in these formulas, particularly in overestimating reliability with small stack sizes, which was already indicated in test series by Chan [13].

Feyen and Gorelick [17] suggested an alternative formulation, which incorporates the variance of log hydraulic conductivity, σ_Y^2, in case of 2D Gaussian distributed random fields: $\langle R \rangle = (S-0.5)/[S+2(\sigma_Y^2+1)]$. Although only demonstrated for the particular case examined, specifically, a water supply problem in a hydroecologically sensitive area, there is clear evidence of the transferability of these findings to other similar cases. A high true reliability may be assigned to solutions found for a rather small number of realizations. For example, for solutions complying with the constraints of a stack size of 100, the approximation of Feyen and Gorelick [17] computes expected true reliabilities $\langle R \rangle$ of over 95% under moderate aquifer heterogeneity ($\sigma_Y^2 = 1$).

For the further demonstration, we use the term "nominal reliability" R_N (cp., [46]), which only states the reliability that is measured for the stack considered. A nominal reliability of $R_N = 100\%$ means that a solution complies with the constraints of all S realizations of the stack. Lower values denote solutions that fail in a certain fraction F of realizations, so that the nominal reliability is expressed by $R_N = (S-F)/S$. As shown by Feyen and Gorelick [17], if the stack size is sufficiently large, the nominal reliability is usually very close to the true reliability. In this work, reliability estimates for a large stack of 500 realizations are supposed to be representative approximations of the true reliability. The latter may be determined or approximated by a post-optimization Monte Carlo analysis of optimal solutions found using a larger stack, i.e., with a huge number of realizations (e.g., [47]). Alternatively, the nominal reliability of a solution may be calculated repeatedly for increasing stack size until it converges to the true reliability (see Fig. 1).

2.2. Optimization procedures

A typical optimal control problem in groundwater management is the adaptation of wells for contaminated site clean-up or water supply. Particularly when considering well placement in heterogeneous aquifers, the related OFs are commonly non-linear and non-convex, exhibiting various local optima [49,4]. Heuristics, such as evolutionary algorithms, prove particularly suitable for solving such demanding problems. However, if the complexity of such problems is maintained at a low level, then classical gradient optimization techniques can be an efficient alternative (e.g., [43,46,17]).

Evolutionary algorithms minimize the risk of getting stuck in potentially existing local optima at the expense of a commonly significant number of iterative OF evaluations. Since each individual OF evaluation processes the results of a groundwater model, the number of iterations must be, as far as possible, reduced to obtain optimized solutions within a reasonable time. For stochastic optimization with multiple realizations, this is even more significant since each OF evaluation requires multiple model runs according to the size of the stack of realizations. Consequently, computational time can be reduced by either decreasing the number of OF evaluations, or by minimizing the number of model calls per OF evaluation.

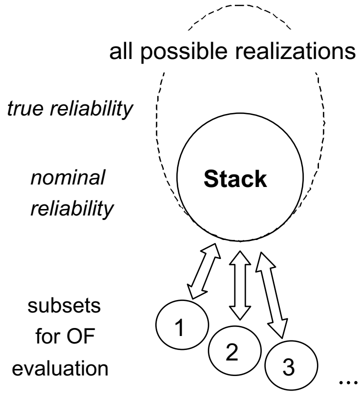

Fig. 1. Definition of true and nominal reliability. A stack of realizations is a representative set of realizations. For reliability-based objective function evaluation, the stack can be considered as a whole, or in small subsets (=evaluation stack).

An evolutionary algorithm, evolution strategies with covariance matrix adaptation (CMA-ES; [22,23]), was selected due to its proven good performance in well optimization problems (e.g., [4]). An appealing feature is the general applicability of this kind of algorithm. CMA-ES is especially suited to multi-modal, complex problems and is regarded to be rather insensitive to noise compared to gradient-based or direct search methods. In [4,5], we implemented this algorithm for solving a moving well problem for hydraulic containment of a contaminated aquifer zone. Both pumping rates and well positions are selected as decision variables. For the water supply problem in this study, an equivalent moving well formulation was used.

In general, evolutionary algorithms encompass a family of generation-based optimization procedures, which are inspired by biological evolution. Accordingly, a specific genetic terminology is used that describes the partly stochastic procedures and operators. For groundwater management optimization, the variants of genetic algorithms (GAs) and evolution strategies have been applied in various studies (e.g., [49,48,28]). Both have a *generation*-based concept in common, that is, instead of evaluating the OF of single solutions iteration by iteration, a set (the *generation*) of potential solutions is considered simultaneously. The first generation may be initialized randomly or deterministically by selecting a number (i.e., population size) of object parameter combinations (i.e., candidate solutions or search points, the *individuals*) from the decision space. For each individual, the OF (the *fitness*) is calculated separately. Then, the individuals of a population are evaluated to produce the next generation (the *offspring*). The fitness (optimality) determines the possibility that an individual is selected to survive, i.e., to be part of the mating pool from which the next generation is produced (*selection*).

Selection means preserving good individuals or, more generally, the information they carry. Thus it is the essential step in converging to an optimal solution. In the next step, two typical operators are introduced to create the next generation: *recombination* and *mutation*. The former is employed to combine the information of different selected individuals. The latter involves a probabilistic change of the selected individuals or their combinations. After creating a new generation of same size, this procedure is repeated until a certain termination criterion (convergence), such as fitness threshold or maximum generation number, is reached.

Compared with the very popular binary GA (e.g., [19]), CMA-ES is a recent and less-known variant of evolutionary algorithms. Without going into the details, we would like to outline the main features of both algorithms. While in binary GA variants the individuals are represented by chromosomes (binary bit strings), CMA-ES uses real-value representations of the individuals. Selection and subsequent recombination are commonly probabilistic in GAs but deterministic in CMA-ES. Consequently, in GAs, higher quality individuals have higher probabilities to survive from one generation to the next. This may even be enforced by elitism, which involves deterministically selecting one or more superior individual from one generation for the next one, without any alteration. CMA-ES uses non-elitist (μ, λ)-selection, that is, the best μ of λ individuals per generation exclusively form the mating pool. A main difference between the two algorithms is the population size (i.e. λ), which has to be set specific to each problem. A preliminary testing of different population sizes is recommended for ideal GA configuration (e.g., [41]), whereas CMA-ES works well with $\lambda = 4 + \lfloor 3 \ln n \rfloor$ and $\mu = \lfloor \lambda/2 \rfloor$ [22]. Parameter n denotes the dimension of the problem, that is, the number of decision variables. In many cases, however, binary GAs behave similarly in converging to an optimum for a broad range of population sizes and these are empirically configured (e.g., [1,4,12]).

Stack ordering adopts a suggestion made in previous studies which deal with GAs for reliability-based design involving multiple realizations [10,44]. The idea is to select only a subset of the stack of realizations during each of the iterations of the optimization algorithm (Fig. 1). This means that a stack of sufficient size is considered throughout the optimization, but only a small number of realizations is sampled for computing the OF value of an individual. Considering the remainder, we will call the entire realization pool "repository stack" and the subset utilized for OF evaluation, "evaluation stack". Although computational time is generally saved by utilizing an evaluation stack of small size, accuracy and exactness are compromised. This is of central concern, particularly due to the potential reliability of solutions analyzed only for a small number of realizations. In fact, in water resource management, high reliabilities are desirable, especially for problems involving the risk of contamination (e.g., [18,44,26]).

Smalley et al. [44] and Gopalakrishnan et al. [21] demonstrate the application of their noisy genetic algorithm for risk-based bioremediation design. The OF is evaluated by taking the average results of a small number of (4–15) realizations (the evaluation stack) that are randomly sampled from the entire stack of realizations. Starting with a very small sample size, algorithm-specific rules are introduced to increase the evaluation stack in later generations. Wu et al. [48] apply this approach for optimizing a sampling network design. As the calculated OF value is an approximation of the true fitness that is governed by the (average over the) respective realizations that are sampled, noise is introduced. Obviously, there is no guarantee that solutions that are valid (not penalized) for the evaluation stack will yield a nominal reliability of $R_{\mathrm{N}} = 100\%$ when being evaluated for the entire stack. However, replacing the evaluation stack at each generation raises the probability that designs which perform well survive, and that unsatisfactory variants are discarded. The suitable size of the evaluation stack and the appropriate configuration of the noisy GA are derived for a specific problem from a computationally intense preliminary survey. After the optimization procedure is terminated, the entire (repository) stack is considered for the calculation of nominal reliabilities.

In their remediation example, the values of R_N span a range between 68% and 98% [21].

Cantoni et al. [10] and Marseguerra et al. [33] present a similar GA implementation for optimal reliability-based design of engineering systems. They also examine possible design configurations by means of evaluation stacks of reduced size, but introduce a dynamic book-keeping procedure to enhance the significance of the OF evaluation. They suggest that the accumulation of evaluation results in an archive, which is continuously updated during the optimization procedure. Thus, the OF value of ‘good’ configurations (i.e., individuals that repeatedly appear in generations) is estimated not only on the basis of the current evaluation stack, but also on the results of previous evaluations of different stacks. Using this “drop-by-drop” approach, statistically more significant results are obtained for good candidate solutions than for less fit ones.

Chan Hilton and Culver [12] introduce a so-called robust genetic algorithm, which only uses evaluation stack sizes of 1. This means that for each generation, one realization is exclusively and randomly sampled for OF evaluation of all individuals. To overcome the significant noise induced by highly realization-specific results, the authors suggest manipulating the common selection scheme in GAs. The focus is placed on those individuals that are fittest and therefore have a high probability of survival in the next generations. Instead of only using fitness as a criterion for selection, an empirical scheme raises the selection probability for individuals according to the number of generations they have already survived (their “age”). Compared with basic GA implementations, the preservation of numerous good individuals over several generations is of essential importance here. In the robust GA [12], this is amplified by a large proportion of elite individuals: only half of the individuals are replaced at each generation.

In [12], the robust GA is judged against a noisy GA implementation for the optimization of the well configuration of a pump-and-treat system in aquifers of low and moderate heterogeneity. The results are obtained for a stack size of 500 realizations and are revealed as highly variable in the OF values and the post-computed nominal reliabilities. In the case of low heterogeneity, nominal reliabilities of solutions range between 76% and 100%. In the moderately heterogeneous case, none of the algorithms provided solutions of more than 80%.

In essence, one can say that noisy or robust GA techniques achieve considerable savings in computation time. Yet, the question persists as to how to save computation time and still arrive at highly reliable and (cost-)optimal configurations. In this study, we will demonstrate that the reliability of a solution stemming from optimization with comparatively small evaluation stacks may be maximized if the focus is on the most “critical” realizations, that is, those realizations that virtually represent the binding constraints.

3. Water supply problem formulation

In the particular problem considered in this study, we adopt the perspective of a decision maker who has to configure both the layout and the protection area of drinking water supply wells. The optimization problem to be set up will answer the question: where should one or more (three, in this case) pumping wells be placed in order to maximize the benefit (i.e., the total groundwater extraction rate) while preventing adverse effects? As for adverse effects, unacceptable depletion of the groundwater table and potential contamination of extracted fresh water are considered.

Formally, the maximization of drinking water extraction from an aquifer is constrained by (i) a strict drawdown threshold over the entire area that is modeled, and by (ii) a given target of minimum travel time to the well(s) from an existing agricultural area located upgradient of the well(s). The latter constraint is inline with common practice in countries with agricultural activities, where well protection areas are delineated with respect to residence time in groundwater (e.g., [9,45]). In these areas, activities threatening the groundwater body are forbidden. The risk emanating from degradable organic compounds and bacteria that enter groundwater outside of these areas is minimized by ensuring a sufficiently long residence time for in-situ degradation.

Formally, the optimization problem is described as follows:

$$\max_{\mathbf{q},\mathbf{i},\mathbf{j}} \frac{\|\mathbf{q}\|_1}{1 + pen(\mathbf{q},\mathbf{i},\mathbf{j})} \tag{1}$$

subject to

$$\mathbf{q}_{\min} \leqslant \mathbf{q} \leqslant \mathbf{q}_{\max} \tag{2a}$$

$$\mathbf{i}_{\min} \leqslant \mathbf{i} \leqslant \mathbf{i}_{\max} \tag{2b}$$

$$\mathbf{j}_{\min} \leqslant \mathbf{j} \leqslant \mathbf{j}_{\max} \tag{2c}$$

where norm $\| \mathbf{q} \|_1$ is the sum of the pumping rates of all wells. Well coordinates are defined by rows $\mathbf{i}$ and columns $\mathbf{j}$, which are subject to optimization within the ranges of a rectangular well placement area as defined by Eqs. (2b) and (2c). The optimization problem is solved within a stack of S realizations that statistically represent the uncertainty of the hydraulic conductivity field. The objective function is similar to that presented by Wagner and Gorelick [47], but incorporates constraint violations for individual realizations as penalties.

In order to achieve high reliability, the goal is to minimize the penalty *pen*, reflecting the most significant constraint violation among all individual realizations, r, of the stack of S realizations. No active constraint yields $pen = 0$, and bears 100% reliability. In this case, the OF value (Eq. (1)) equals the total pumping rate to be maximized. The two constraints on drawdown and travel time of groundwater from potentially contaminated zones are combined in *pen* as exponential expressions:

$$pen(\mathbf{q},\mathbf{i},\mathbf{j}) = \max{}_r A^{d_r(\mathbf{q},\mathbf{i},\mathbf{j})} + \max{}_r B^{z_r(\mathbf{q},\mathbf{i},\mathbf{j})} \tag{3}$$

where d_r quantifies the drawdown constraint violation for each realization r:

$$d_r(\mathbf{q},\mathbf{i},\mathbf{j}) = \max\left[0, \frac{d_{\mathrm{act},r}(\mathbf{q},\mathbf{i},\mathbf{j}) - d_{\max}}{d_{\max}}\right] \tag{4}$$

and, equivalently, z_r refers to unacceptable residence time of the groundwater:

$$z_r(\mathbf{q},\mathbf{i},\mathbf{j}) = \max\left[0, \frac{z_{\min} - z_{\mathrm{act},r}(\mathbf{q},\mathbf{i},\mathbf{j})}{z_{\min}}\right] \tag{5}$$

$$r = 1,\ldots,S \tag{6}$$

The exponents in Eq. (3) are higher than zero when realization-specific results surmount the threshold on drawdown, $d_{\max}$, or residence time is below the given minimum of $z_{\min}$. The drawdown term $d_{\mathrm{act},r}$ is quantified individually for each realization r as follows: First the head change, i.e., the drawdown of the water table due to pumping for a well configuration $(\mathbf{q},\mathbf{i},\mathbf{j})$ is computed by comparison to heads for undisturbed flow. Then $d_{\mathrm{act},r}$ is calculated as the highest measured relative drawdown in the model domain, which certainly occurs at one of the pumping well positions. A penalty $(d_r > 0)$ is assigned, when $d_{\mathrm{act},r}$ exceeds a given problem-specific drawdown limit $d_{\max}$, which is set fixed for all realizations (Eq. (4)).

In terms of residence time, the risk of contamination relies on the assumption that solute transport in groundwater occurs exclusively by advection, that is, no hydrodynamic dispersion is taken into account. The penalty on inadequate residence time is a function of the lowest residence time of all particles, $z_{\mathrm{act},r}$. The latter is obtained by forward tracking of particles (using MODPATH, [39]) originating from potentially contaminated zones to the wells of a tested well configuration. A penalty $(z_r > 0)$ is assigned if a given travel time threshold $z_{\min}$ is violated (Eq. (5)). By setting A and B to 10^{100}, the value of pen already becomes 10 when a relative violation of 1% for one constraint is found (Eq. (3)), and then exponentially rises for more severe violations. Slightly invalid candidate solutions are thus assigned an OF value that is at least one order of magnitude higher than the total pumping rate. Note that for the specification of A and B, as given above, a precursory sensitivity analysis was carried to examine their effect and to find out appropriate values for the optimization problem.

In general, the presented penalty concept is only one example of many possible variants. The effect on the optimization procedure should be the same for any other approach as long as invalid solutions are found undesirable due to penalized OF values, and as long as the penalty reflects the degree of violation for both constraints. This is because, in each generation, CMA-ES selection of parents is rank-based, and therefore only the comparative fitness values among individuals, and not their absolute OF values, are relevant. In other words, this means that selection is scaling invariant.

4. Demonstration example

To demonstrate the stochastic optimization procedure, we employ a synthetic groundwater model set-up. A 3D finite difference model implemented in MODFLOW 2000 [24] is used to simulate an unconfined aquifer consisting of two connected sedimentary layers. The model area is discretized into 100 rows and 150 columns of $10 \times 10\ \mathrm{m}^2$. The thickness of both layers is set to 10 m. By using constant head boundaries at the east (head: 17.75 m), and west (20 m), and no flow boundaries on the north and south, a regional hydraulic gradient of 0.15% was obtained. A recharge of 365 mm/yr is assumed for the entire model domain. The top layer is a mixture of fine sand ($k = 5\mathrm{e}{-4}$ m/s, volumetric portion = 70%) and gravel (1e−2 m/s, 30%); the bottom layer consists of fine sand (5e−4 m/s, 40%), coarse sand (1e−3 m/s, 40%) and gravel (1e−2 m/s, 20%). Indicator kriging is used to produce the distribution of hydraulic conductivity of both layers based on exponential variograms [15]. Correlation lengths are set to 300 m in a N–S direction, and 150 m in E–W. There is no correlation between the hydrofacies in both layers, simulating a sedimentary stratigraphy that originates from two depositional events (Fig. 2). Fig. 3 depicts a plan view of the flow domain, with hydraulic head isolines of the top layer under undisturbed conditions (no well active).

The unconditioned synthetic base case is generated to define a virtual reality, i.e. the truth, which is assumed to

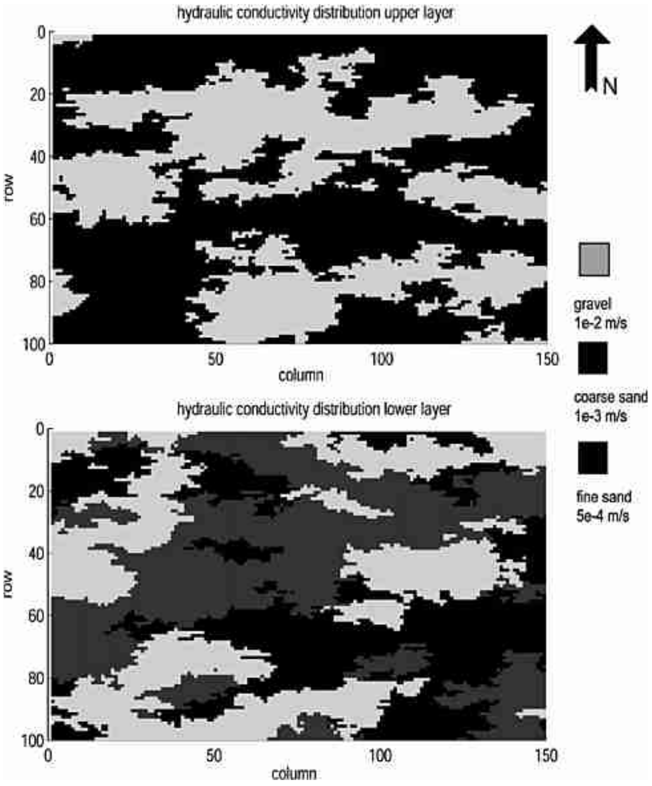

Fig. 2. Hydraulic conductivity distribution of top and bottom layers for base case. The row and column sizes of the uniform grid are 10 m wide.

Please cite this article in press as: Bayer P et al., Computationally efficient stochastic optimization using ..., Adv Water Resour (2007), doi:10.1016/j.advwatres.2007.09.004

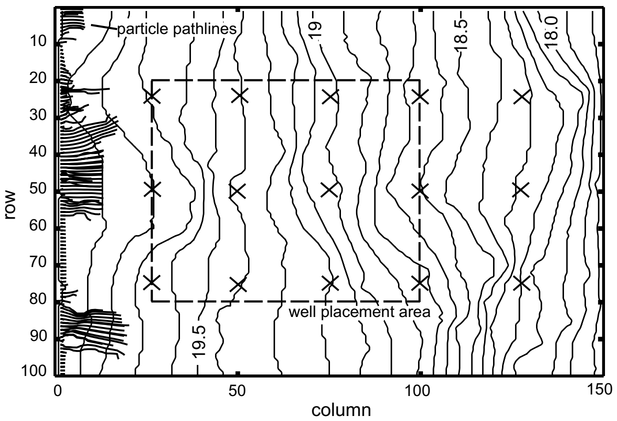

Fig. 3. Top view of base case hydraulic model with head isolines and 50 d pathlines of particles emanating from the western upgradient boundary. Crosses denote measurement points used to generate conditioned aquifer realizations.

be only locally known. Conductivities in both layers and heads from undisturbed, steady state modeling are sampled at 15 locations as defined by a uniform 25 m mesh (see crosses in Fig. 3).

The sampled conductivities are then used to produce conditioned conductivity maps for both layers. Again, indicator kriging is applied, however, with variable, randomly chosen exponential variogram settings. In particular, volumetric portions of the different sedimentary facies and their correlation lengths are not supposed to be exactly known within a range of 25% around the values chosen for the base case. The same uncertainty interval of 25% is selected for the recharge parameter, which was identified as a considerable factor in the planning of wellhead protection programs by Jyrkama and Sykes [27]. Undisturbed flow is modeled for each realization, and the simulated heads at the measurement points are compared to those sampled in the true base case. Only those realizations that closely match the base case are retained (matching criterion: absolute head difference $\leqslant 0.1$ m).

As an additional criterion, the total water volume inflow rate of the candidate model aquifers is compared to that computed for the base case. All candidate aquifers that derived differences of more than 10% compared to the base case are dropped. In practice, such comprehensive knowledge of the regional water budget may not be available. For this hypothetical example, this procedure is applied to produce aquifer realizations that differ only marginally in their total water budget and so omit extreme outliers that incessantly control highly reliable solutions. It can be compared to defining a prescribed flow condition at the western boundary. This procedure is repeated until a stack of 500 equally probable aquifer realizations is created (i.e. in this case, stopped after approx. 375.000 iterations). Obviously, this trial-and-error approach may be replaced by alternative inverse modeling methods. Due to the low computational time for generating and comparing new realizations, we preferred to use this straightforward, brute force type of method (cp. [7]).

For the optimization problem, the drawdown constraint $d_{\max}$ is set at 1 m (Eq. (5)) over the entire domain. Moreover, it is supposed that pollutants can only emanate from sources beyond the upgradient model boundary. This is simulated by tracking particles that originate from the western boundary grid cells. By placing one particle in each cell of both layers, a total of 200 particle paths are examined. The respective threshold of groundwater residence time, $z_{\min}$ (Eq. (6)), is set to 50 days, according to common standards in Germany. For each candidate solution, particle travel times to the wells are computed for any realization of the evaluation stack and realization-specific lowest values (for the fastest particle), $z_{\mathrm{act},r}$ are determined assuming steady-state conditions. Fig. 3 presents a top view of the particle flowpaths at undisturbed flow conditions within 50 days.

We examine single-well and 3-well problems with free well positions within a well placement area, that is situated downgradient of the particle paths ($\mathbf{i}_{\min} = 20$, $\mathbf{i}_{\max} = 80$, $\mathbf{j}_{\min} = 25$, $\mathbf{j}_{\max} = 100$). Well positions are assigned to the nearest neighbor grid cell of the real valued location suggested by the optimization algorithm. The range of acceptable pumping rates at individual wells is set at $\mathbf{q}_{\min} = 5$ l/s and $\mathbf{q}_{\max} = 50$ l/s for all cases (following a preliminary investigation with some trial runs). For further discussion on the implementation of an advective control problem that is solved by evolutionary algorithms, the interested reader is referred to Bayer and Finkel [4].

5. Stack ordering procedures

5.1. Basic stack ordering (SO) and no ordering (NO) procedure

The innovative element in the stack ordering procedure presented in this paper is a learning method which, in contrast to the common random sampling approach, guides successive selection of realizations. Each time an individual solution (pumping scheme) is evaluated for a stack of realizations, there will be one or a few realizations that produce the worst results, showing the highest violations of constraints or, more generally, producing the lowest OF values. However, focusing only on a small number (i.e., a fixed set) of critical realizations for optimizing the OF, instead of the entire stack, is not recommendable for non-linear, non-convex problems, such as the moving well formulation. This is because the most critical realizations will change depending on the particular combination of decision variables (i.e., location of wells and corresponding pumping rates) of the evaluated solutions. At different well positions different realizations represent the binding constraints. Therefore, available approaches for the detection of "worst case" realizations (e.g., [40,20,31,32]) can assist in finding extreme realizations for a given well layout, but are hardly sufficient in identifying the critical realization for the entire design optimization problem. In other words, as long as the 100% optimum (that is, the pumping scheme that maximizes the OF while meeting the constraints for all realizations) for a stack is not known, it is virtually impossible to interpret the relevance of extreme model responses for this optimum.

During iterative optimization, such as evolutionary search, the decision space is roughly investigated at the outset, so that significantly distinct candidate solutions are evaluated in the first generations. When converging to an optimum, exploration of decision space abates and local search evolves until the algorithm converges to a (locally) optimal or near-optimal solution. The disparity among the candidate solutions in the initial phase yields a variety of critical realizations, whereas in a later phase, local search is governed by a small number of realizations that represent the binding constraints for (locally) optimal and near-optimal solutions. Hence, it seems reasonable to combine search for optimum and detection of critical realizations directly.

A direct procedure is to sample one realization after another, and then to ‘‘earmark’’ the one that yields the worst result. Alternatively, in particular for cases where optimization is dominated by strict design criteria (i.e., in terms of constraints implemented in the OF as penalties), those realizations which yield penalized results may be branded. In order to exploit information about the potentially critical nature of realizations for the optimization procedure, we suggest assigning a credit to the (‘‘bad’’) realizations and conducting an ordered examination of the evaluation stack, starting with the realization that has the highest credit. The sequence of realizations that are sampled for the evaluation of the OF is continuously updated. If a penalized OF value appears, a new credit is added to the credit account of the respective realization according to the position of the stack. Before the next iteration (evaluation of next individual) starts, the stack is sorted according to the credits. The desired consequence is that critical realizations accumulate on top of the stack and are evaluated first.

Merely assigning credits and sorting the stack, though helpful in detecting potentially critical realizations, does not provide the benefit of saving computational time. For this purpose, we suggest stopping the OF evaluation if a critical realization is discovered (Fig. 4a). Since the current candidate solution represents an invalid design and will hardly be the 100% reliable optimum, no exact OF calculation is required. Stopping before the entire stack has been evaluated obviously yields an approximate, potentially underestimated value of the OF, as further critical realizations can be expected in the remaining part of the stack which has not been evaluated. This means we introduce bias, and thus noise, by breaking. The pertinent question is whether this significantly compromises the performance of the search algorithm.

In view of the robustness of evolutionary algorithms with respect to a certain degree of noise, stopping should not represent a shortcoming. Furthermore, noise is only introduced for invalid designs, and thus is expected to be less influential compared with approximating feasible solutions. Finally, the combination of stack ordering and breaking early continuously collects critical realizations that will be evaluated first in subsequent iterations. As a result, the closer the optimization converges to a solution, the more mature the set of critical realizations becomes, and thus more accurate is the OF evaluation.

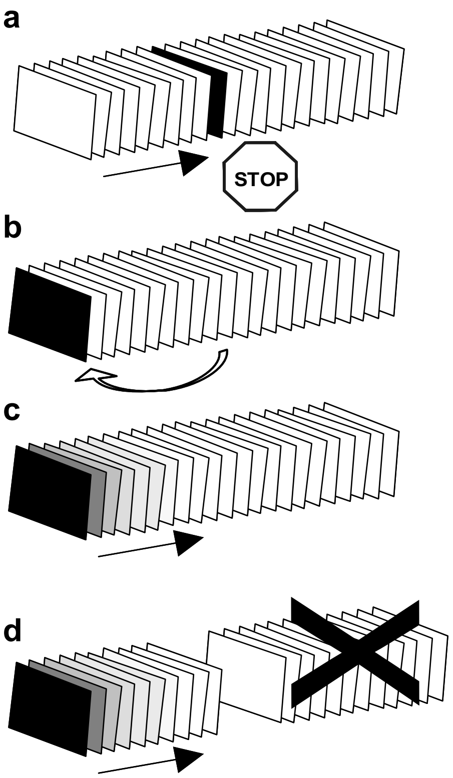

Fig. 4. Sketch illustrating basic stack ordering procedure (SO, a–c) and stack ordering with reduction (SORed, d). Starting from top of the stack (left side), realizations are evaluated until a certain criterion is not fulfilled, for example, a penalized solution is calculated. The procedure is then stopped, the entire stack is randomized and sorted according to the credits (b), a process which is iteratively repeated (c). If, after numerous iterations no new critical realizations are found, non-penalized realizations are only probabilistically sampled or, as depicted in (d), discarded as a whole.

The general procedure of the basic stack ordering and break (SO) method is schematically illustrated in Figs. 4a–c. Consider an initially unsorted stack (Fig. 4a). Iterations (over potential pumping schemes) of the optimization are counted from $t = 1, \ldots, t_{\max}$. Note that for an evolutionary algorithm, iterations are equivalent to individuals. If, during the sampling of realizations, the currently evaluated solution does meet design constraint Ω (i.e., it is penalized) for a realization r, then further sampling is stopped. The number of realizations considered in iteration t may be smaller than the size of the stack, and equals the position of realization r in the stack, hereafter denoted as $s_r(t)$ (with $1 \leqslant s_r(t) \leqslant S$). The OF is approximated accordingly, as realizations with later positions in the stack may yield a higher penalty, but are not considered. A credit C_r is assigned to the corresponding realization r, which is

$$C_r(t+1) = C_r(t) + \sqrt{s_r(t) - 1}. \tag{7}$$

This is done sequentially for each iteration (i.e., pumping scheme). Note that taking the square root instead of just the position leads to a more pronounced distinction between the top positions in the stack. In contrast, realizations which appear at much later positions are almost credited the same. The rationale behind this is that the position of realizations without any credits is not ranked and these realizations appear at bottom of the stack. It is suggested that the list of un-credited realizations be randomized before each new ranking phase in order to minimize the influence of the initial (random) stack order.

The following simple example clarifies the SO procedure: Consider a small stack size of $S = 22$ realizations r ($r = 1, \ldots, 22$). Initially, as no credits have yet been assigned, $C_r(t = 0) = 0$ for each r, and the position of each realization in the stack is $s_r(t = 0) = r$. The optimization is started with the first individual (i.e., well configuration), indexed by $t = 1$. First, the model is run for realization at position $s_r = 1$, and no constraint violation is found. This is repeated for the next realizations until realization at position $s_r = 10$ yields a penalty (Fig. 4a). At this point, further testing of realizations is stopped and the OF is assigned the (penalized) value calculated for realization $r = 10$ at $s_{10}(t = 1) = 10$. Before repeating this procedure for the next individual ($t = 2$), realization $r = 10$ is assigned a credit of $C_{10}(t = 2) = 3$ (Eq. (7)). After that, the stack is shuffled and then ordered according to the credits assigned so far. This means realization $r = 10$ moves to the top of the stack (Fig. 4b): $s_{10}(t = 2) = 1$ and the other realizations' position is randomized.

For the subsequent individual $t = 2$ the procedure described above is repeated until a certain realization r violates the constraints, gets a credit according to its position $s_r(t = 2)$ and a new position in the stack for $t = 3$. It will displace realization 10 from the top of the stack if $s_r(t = 2) > 10$ since $C_r(t = 3) > C_{10}(t = 3) = 3$. Please note that no credit will be assigned to realization $r = 10$ (in case it violates the constraints again), due to its top position (Eq. (7)). Also no credit is assigned if no constraint violation is found after testing of all realizations. During the course of the optimization, iteration by iteration, an increasing number of critical realizations will be identified and ranked within the stack according to their credits.

The basic SO procedure is applied to the water supply problem and compared to a so-called "NO procedure". The latter is similarly based on an examination of each realization for each OF evaluation, but uses an entirely randomized stack. The comparison is to reveal the efficiency of stack ordering. In addition to the basic SO procedure, subsequent stack ordering variants (SORed, SORep, SORepDecay) are developed. A summary and comparison to random realization sampling methods (NO, NORep) is given in Fig. 5.

5.2. The stack ordering and reduction (SORed) procedure

From experience and knowledge gained while applying the SO method, we further developed the method so as to increase the efficiency of the optimization procedure. We call this new method "stack ordering and reduction" (SORed) (Figs. 4 and 5) whereas "reduction" refers to the probability of sampling. The procedure works like SO as long as new critical realizations are found more or less regularly from generation to generation. The modification only becomes active if no new critical realizations are found in f previous generations, that is, if no previously uncredited realization has been assigned a credit. Parameter f represents the iteration threshold. This also means that no savings could be made in previous generations since no break occurred. If so, a reduced probability of sampling is introduced, causing not all realizations to be sampled anymore. By defining an additional criterion C^* (with $C^* \geqslant 1$), the probability is determined as follows. For realizations with very low or no credits ($C_r < C^* - 1$) sampling probability is $p_r = (1 + C_r)/C^*$, whereas those realizations with $C_r \geqslant C^* - 1$ are definitely selected ($p_r = 1$). Computational savings decrease the closer C^* is set to its lower limit 1. In essence, those realizations which have so far significantly controlled the evolutionary search are most likely to be considered, while others are likely to be discarded in order to save model runs.

The iteration threshold f is set as a multiple of generations, reflecting that major changes of individuals have to be expected from one generation to the next. In the presented example application, two values of f are considered, $f = 5$ and $f = 10$, in order to examine the influence of this parameter on the outcome. These values of f represent thresholds of ($\lambda \times f = 7 \times 5=$) 35 or ($7 \times 10=$) 70 iterations for the single well case, and of ($\lambda \times f = 10 \times 5=$) 50 or ($10 \times 10=$) 100 iterations for the 3-well case. The value of C^* is set to 4 and so all non-credited realizations are sampled according to probability 1/4. Although this value is empirically based on a preliminary investigation, sampling only every fourth is assumed to be a very conservative estimate, and even rigorously neglecting all non-credited realizations could lead to satisfactory results. However, as soon as some realizations of the stack are, potentially, not sampled, the possibility of detecting solutions having a nominal reliability R_{N} of less than 100% will increase. The higher the value of C^* is set, the higher this chance is. In view of this, C^* should be as low as possible, but in practice will mainly be limited by the expected computational effort. In this study, the values of R_{N} are computed in a post-optimization investigation by testing a suggested (optimized) well configuration for the entire stack.

5.3. Stack ordering and no ordering with replacement (SORep, SORepDecay and NORep)

This alternative implementation of stack ordering adopts suggestions by previous studies (e.g., [44,10]) to distinguish between a small evaluation stack and storing the remaining realizations in the repository stack. This means inspecting only a subset of the 500 realizations for the evaluation of candidate solutions (see Fig. 1). A self-evident procedure is to utilize only highly credited realizations in

Instruction	Free parameters	SO	SORed	SORep	SORep Decay	NO	NORep
Evaluate entire stack		○	○	○	○	⊗	○
Stack ordering and break; assign credit C_r to critical realization r according to position of the stack s_r $C_r(t+1) = C_r(t) + \sqrt{s_r(t) - 1}$		⊗	⊗	⊗	⊗	○	○
Start with probabilistic sampling of realizations with probability p_r $C_r > C^*$: $p_r = 1$ $C_r \leq C^*$: $p_r = (1+C_r)/C^*$ if no new critical realizations found in f previous generations	f, C*	○	⊗	○	○	○	○
Only probabilistic sampling of realizations with probability p_r $C_r > C^*$: $p_r = 1$ $C_r \leq C^*$: $p_r = (1+C_r)/C^*$ for generating evaluation stack of fixed size S_{eval}	S_{eval}, C*	○	○	⊗	○	○	○
Decay of credits $C_r = k\,C_r$ if no credit assigned in previous iteration	S_{eval}, C*, k	○	○	○	⊗	○	○
Random sampling for generating evaluation stack of fixed size S_{eval}	S_{eval}	○	○	○	○	○	⊗

Fig. 5. Procedural elements and related free parameters of stack ordering and random sampling methods.

the evaluation stack. Such a procedure is comparable to SORed with the difference that none of the non-credited realizations is sampled. However, if doing so, one major question is: which realizations are the most critical ones? Since these realizations are not known at the start of an OR, we again propose dynamically identifying them during the search. An ideal method of achieving this is by assigning credits, as demonstrated above.

As the most basic implementation, we define a size, S_{eval}, of the evaluation stack that is fixed within an optimization procedure (Fig. 6). The evaluation stack represents a subset of the 500 realizations that is updated for each new individual. In the beginning of the optimization, no credits are assigned yet, and S_{eval} realizations are sampled randomly. As soon as credited realizations are encountered, probability-based sampling, as introduced for SORed, is applied. Note that no iteration threshold *f* is used here. Equivalently to SO and SORed, all 500 realizations are randomized and then sorted according to their credits. Starting from the top of the entire stack, the evaluation stack is filled up with S_{eval} realizations. This is done prior to each iteration. That is, the evaluation stack used for the preceding iteration is replaced by a new one. Accordingly, this procedure is termed "stack ordering with replacement" (SORep). The rest of the realizations are kept in the repository stack, and then merged with the evaluation stack for the successive randomizing, sorting and sampling event.

Subsequently, "stack ordering with replacement and decay" (SORepDecay) is to be developed. Here, a decay

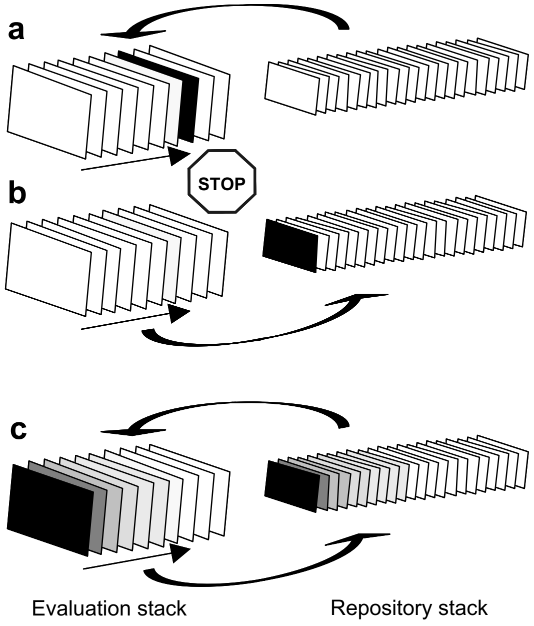

Fig. 6. Sketch illustrating stack ordering with two subsets, an evaluation and a repository stack, which is realized by SORep and SORepDecay. Starting with a random selection of realizations take from the entire stack, the evaluation stack (left side) is sampled until a certain criterion is not fulfilled, for example a penalized solution is calculated (a). Then the procedure is stopped, the evaluation stack is merged with the remaining realizations in the repository stack, and the entire stack is randomized, then sorted according to the credits (b), a process which is iteratively repeated (c).

factor is introduced that decreases the credits of realizations during the optimization. It builds upon the results of the other stack ordering procedures and, hence, will be described in detail in the following section.

For the purpose of comparison, we also employed NORep, which stands for no ordering, and random sampling of realizations for evaluation. This conceptualizes the procedure of a noisy GA, but with fixed evaluation stack size (e.g., [44]). The comparison will show the effect of stack ordering, particularly with respect to optimality and reliability of the optimized results.

6. Application and results

6.1. Optimization procedure and evaluation

Due to the stochastic nature of evolutionary algorithms, applying more than one optimization run (OR) is recommended in order to raise the probability of detecting the global optimum. One OR corresponds to one CMA-ES application with a fixed number of objective function evaluations (i.e., iterations). Each OR is unique and yields different convergence characteristics when converging to an optimum. Therefore, representative results are only obtained by inspecting several ORs. When using time-consuming numerical models, a satisfactory and statistically correct examination of the optimization procedure is restricted by the high computational effort that would be required for dozens to hundreds of ORs and a stack size of 500 realizations. As a compromise, it is common practice to use a feasible number of ORs for each problem or setting, which would enable insight into the major characteristics (convergence rate) of the evolutionary search. In subsequent applications, statistical results are presented. Depending on the problem (i.e., well case) to be solved, for repeated ORs the number of OF evaluations (i.e., individuals) is predefined and not varied. Median convergence rates as well as the 10% and 90% quantiles are shown for 10 ORs (options SO, SORed) and 25 ORs (options SORep, SORepDecay), respectively. This is in line with related work in this field (e.g., [34,8]).

In a previous study on well placement for pump-and-treat-systems with 100 different deterministic heterogeneous models, Bayer and Finkel (2007) recommended the use of about 50 generations per well. Comparatively conservative settings were chosen here. The maximum number of iterations was set to 602 (i.e., 86 generations, $\lambda = 7$) per OR for the single-well case and 1800 (180 generations, number of individuals per generation $\lambda = 10$) for the 3-well case. Please note that each well is represented by three decision variables: the pumping rate and the two coordinates.

Both extraction rates of single-well and 3-well configurations have been maximized. The several hundred ORs performed per well configuration delivered numerous different optimized systems, reflecting both the complexity of the problem being solved as well as the singularity of CMA-ES runs. As a full enumeration of the decision space was computationally unfeasible, it cannot be assured that, even after this exhaustive examination, the globally optimal well configurations were found. Nevertheless, the best 100% reliable solutions that had been detected were reproduced several times (same or similar well positions and slightly different pumping rates). Consequently, it can be assumed that these are either very close or equal to the global optima, and hence we use the fitness of these solutions as references for assessment of optimized well configurations.

Table 1 lists these reference well layouts, and exposes a common feature: preferable well positions appear to be close to those measurement locations that indicate high hydraulic conductivity values and served as conditioning points for generating the stack of realizations. These positions appear to be particularly optimal since here modeling results show the least variations. Therefore, for the test case considered, this observation can even be regarded as common sense criterion to assess the optimality of well positions, since the global optima are not known. Particularly at a central (row 51, column 50) and a northwestern location (25/75) in the well placement area, high pumping rates can be achieved due to comparably high local conductivities and due to a certain distance from the potentially contaminated upgradient model boundary. The two best well locations identified are very close or exactly match with measurement locations. Ideal 3-well systems share the first position (51/50). Additional wells commonly have similar pumping rates and are, again, favorably located close to measurement points. Three optimized variants with considerably different geometric arrangement of wells are listed in Table 1. Compared to the outcome for the single-well case, the total extraction rate can be more than doubled by adding two further wells.

6.2. Stack ordering and break (SO)

We start with the analysis of the single-well case, which is optimized by evaluating the entire stack without ordering (NO) and with the presented basic stack ordering (SO) procedure. As listed in Table 2, comparable values are obtained for median fitness values after 602 function evaluations, which lie slightly below the optimum (see Table 1). Apparently, a fraction of the 10 ORs has not converged to the desired solution (minimum $q = 32.4$ l/s), which reflects

Table 1
Best results for single-well and 3-well problems which are assumed to be global or close-optimal solutions

q	Row	Col.							
1-Well case									
32.4	25	75							
32.3	51	50							
$\sum q$	q_1	Row	Col.	q_2	Row	Col.	q_3	Row	Col.
3-Well case									
67.7	24.7	50	49	27.8	25	74	15.2	77	39
67.4	23.9	50	50	18.9	25	95	24.6	25	51
66.5	24.3	50	51	20.9	26	99	21.3	75	49

Table 2
Single-well case: statistics of fitness values, reliability, savings and average number of critical realizations for NO, SO, and SORed

	NO	SO	SORed ($f = 10$)	SORed ($f = 5$)
Median fitness, q (l/s)	30.8	31.1	30.9	30.8
10% quantile (l/s)	27.4	28.2	27.8	28.2
90% quantile (l/s)	32.4	32.4	32.4	32.4
Median reliability (%)	100	100	100	100
10% quantile (%)	100	100	99.8	100
90% quantile (%)	100	100	100	100
Median savings ($t = 300$, %)	0	35.1	37.9	47.8
10% quantile (%)	0	28.6	29.8	37.4
90% quantile (%)	0	39.6	48.0	57.4
Median savings ($t = 602$, %)	0	39.9	58.5	66.1
10% quantile (%)	0	36.7	49.3	59.8
90% quantile (%)	0	46.4	66.2	72.0
Median number of credited realizations	0	36	32	36

For the latter, two variants are examined with two different values of the threshold parameter f for the switching point to probabilistic sampling. Savings are calculated by comparing the total number of model runs to the number that would be established if all realizations were used.

the difficulty in solving the moving well problem for even one single well, as well as the limited search ability of heuristic optimization algorithms. Specifically, 2 solutions found by NO and 3 solutions by SO are wells far away from the supposed optima listed in Table 1. However, a paramount finding is the similarity of statistics for the convergence properties of both approaches (Table 2, Fig. 7). This suggests that the noise introduced by SO does not significantly corrupt the evolutionary search here.

Due to the immense computational burden, the 3-well problem was not solved by NO. The results for the SO approach are listed in Table 3. Again, each CMA-ES run shows individual characteristics, and as reflected by the lower 10% quantile, some of the identified well configurations are sub-optimal. Despite this, the median fitness of the best solutions after 1800 iterations (Table 3) converged very closely to the desired values listed in Table 1. The median total pumping rate, $\sum q$, reached 60 l/s, which is about 10% lower than the overall best solution found for this problem. Further inspection of individual ORs suggests that even 1800 function evaluations were not sufficient to converge to proximity of the optimal solution for about a half of the ORs. The issue of premature termination, which may prevent the solution procedure from identifying a satisfactory well configuration, is of essential importance in practice. A common approach in practical applications is, therefore, to consider a termination criterion that is oriented at the course of the optimization, and stops the OR if, for example, some stagnation criterion is met.

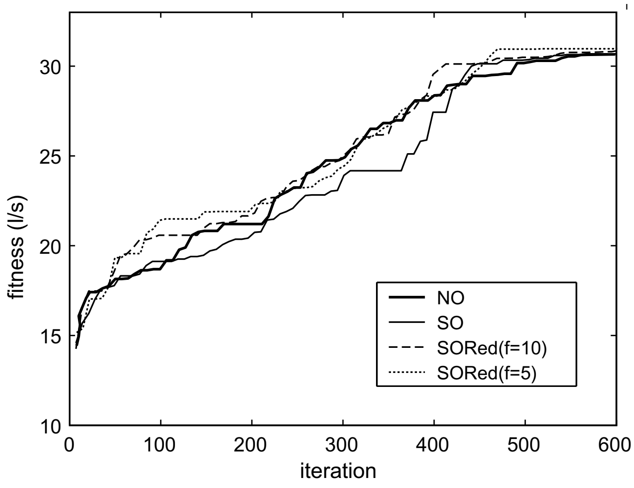

Fig. 7. Median convergences of 10 optimization runs (ORs) for single-well problem for no ordering (NO), and three stack ordering methods.

Within this study, emphasis is placed on the comparison between different methods of stack ordering with a conventional multiple realizations approach. In order to achieve direct comparability, we set a fixed maximum number of iterations for all ORs, independent of the particular solution method used. This was done even if individual ORs may not have reached a satisfactory convergence, and in this way, we guaranteed the same conditions for all methods we examined. Restricting the computational timeframe is also in accordance with related work, for example by Smalley et al. [44] and Chan Hilton and Culver [12].

The typical history of an OR when SO is applied is shown in Figs. 8a and b for both the single- and the 3-well case, respectively. In both well cases, a general observation is that the number of credited (=critical) realizations increases, particularly during the initial search phase, and then stays nearly constant. This is because the search becomes more and more local. During the later search phase, (most of) the realizations that determine the structure of the respective part of the fitness landscape are already grouped on top of the stack. In other words, the more mature the search, the less important the role of non-credited realizations is. However, as local search evolves and "moves", critical realizations can still be found, even at later stages.

Please cite this article in press as: Bayer P et al., Computationally efficient stochastic optimization using ..., Adv Water Resour (2007), doi:10.1016/j.advwatres.2007.09.004

Table 3
3-Well case: statistics of results for two stack ordering implementations, SO (stack ordering) and SORed (stack ordering and reduction)

	SO	SORed ($f = 10$)	SORed ($f = 5$)
Median fitness, $\sum q$ (l/s)	60.0	61.6	60.7
10% quantile (l/s)	58.0	56.8	58.2
90% quantile (l/s)	65.3	66.3	63.7
Median reliability (%)	100	100	100
10% quantile (%)	100	99.8	99.8
90% quantile (%)	100	100	100
Median savings ($t = 900$, %)	48.7	56.0	71.3
10% quantile (%)	43.9	49.6	65.3
90% quantile (%)	53.0	61.6	74.7
Median savings ($t = 1800$, %)	51.0	69.3	78.5
10% quantile (%)	47.5	61.3	75.3
90% quantile (%)	55.4	73.3	81.0
Median number of credited r	62	57	60

For the latter, two variants are examined with different values of the threshold parameter f for the switching point to probabilistic sampling. Savings are calculated by comparing the total number of model runs to the number that would occur if all realizations were used.

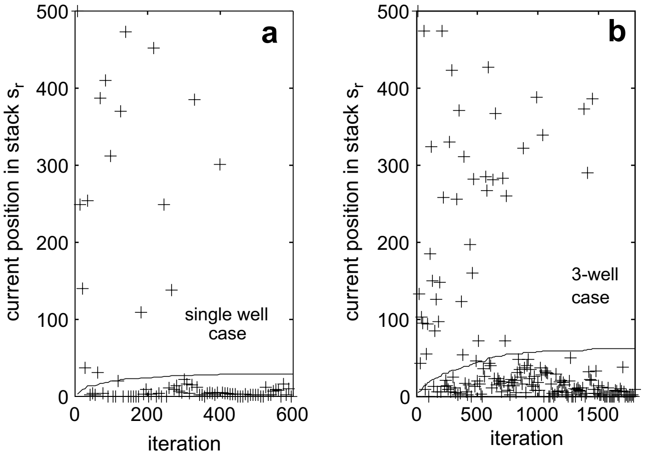

Fig. 8. Positions of new credited realizations (cross markers) and number of credited realizations in the sorted stack (lines) for two optimization runs using SO applied to (a) single-well case and (b) 3-well case.

The absolute savings with SO are very similar for all ORs, as reported by the statistics in Tables 2 and 3. About 40% of the model runs could be saved after 602 iterations for the single well problem. An even higher value of 51% was observed for the 3-well case after 1800 iterations. The more demanding search investigates an elevated number of invalid (i.e., penalized solutions) and so causes a break in further sampling with increased frequency. Accordingly, the (median) number of credited realizations is higher for the higher-dimensional problem, which rises from about 35 to 60.

6.3. Stack ordering and reduction (SORed)

The findings shown in Fig. 8 inspired the set-up of the SORed variant. If during the course of the optimization the number of credited realizations stagnates, why not simply neglecting the rest of the stack? In accordance with the NO and SO approaches described above, 10 ORs were conducted for each problem and value of f. The statistics reported in Tables 2 and 3 highlight the similar statistics for the optimized fitness values, which basically proves comparable performance of the solver for all implementations with different stack ordering variants (Fig. 7). The new finding for SORed is that, as anticipated, occasionally nominal reliabilities R_N of the optimized solutions do not reach 100%. However, median reliabilities are maximal, and critical realizations were overlooked in only a few number of optimization runs. This is quantified by the 90% quantiles of R_N, independent of the number of wells and the value of f. Maximally, one relevant realization was neglected ($R_N = 99.8\%$).

Of particular interest are the computational savings in model runs that can be achieved by ordering the stack. Fig. 9 contains the average (median) relative savings as function of the number of iterations for the two variants of the SORed method in comparison to SO. In the initial search phase of both well problems, savings tend to be very high, then decrease and remain almost constant for SO. Initial savings result from early breaks of the evaluation as invalid solutions appear frequently during this less guided global search phase. Also, in this early phase, the savings trade-offs for all stack ordering implementations are predominantly irregular. This is due to random initialization and the initial explorative search.

For the SORed variants, star markers denote the switching-points when probability-based sampling comes into play. As soon as switching takes place, a significant increase in savings can be observed. As shown in Fig. 9b, savings tend to level off in the long run. The uniqueness of evolutionary search yields a scattered picture of switching points. As expected, for $f = 5$, switching occurs earlier after about 35 generations on the average in both well cases. At $f = 10$, switching points are concentrated at 350 iterations (50 generations) for the single-well case, and

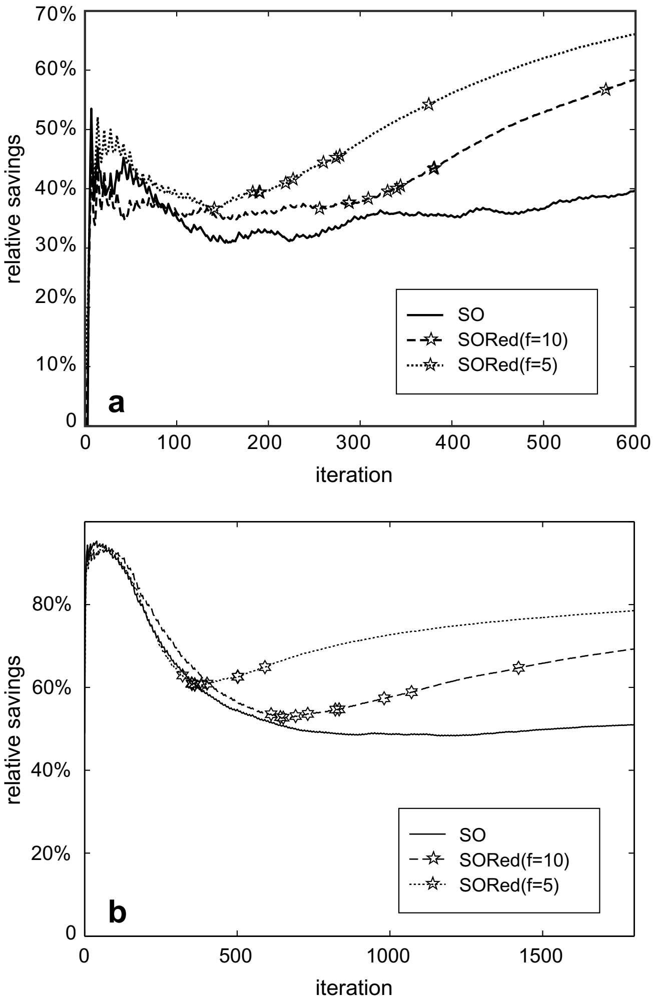

Fig. 9. Median savings of 10 optimization runs for (a) single-well problem and (b) the 3-well problem for two stack ordering methods, SO and SORed. For the latter, two variants are examined with different values of the threshold parameter *f* for the switching point to probabilistic sampling. Stars denote the OR-specific point where switching to probability-based selection occurs.

for the 3-well case, about twice the number of iterations (70 generations) is required. As a general conclusion we can state that high values of *f* and problem dimension tend to delay switching.

Comparing the number of credited realizations of the different stack ordering methods shows these median values to be nearly constant (Tables 2 and 3). This suggests that the number of credited realizations is problem-dependent, but not controlled by the different solution procedures.

Probability-based sampling increases savings, particularly when the smaller value of *f* is chosen. However, higher differences exist between the individual ORs as an effect of the different switching point positions. This is quantified by the quantiles for savings at halftime ($t = t_{\max}/2$) and after the entire number of iterations in Tables 2 and 3. Although computational savings of between 60% and 80% are significant, solving such a multiple realization problem can still require an enormous computational effort. For the problem presented here, 100,000 to 200,000 model runs are needed for one OR, which would become prohibitive if the model running time were to take longer than seconds. Therefore, more rigorous stack ordering methods are proposed in the next chapters.

6.4. Stack ordering and replacement (SORep)

For this stack ordering variant, the objective function is only evaluated by using a small subset of all realizations, that is, the evaluation stack. It has a fixed size of S_{eval} realizations that are dynamically replaced with better odds of achieving those realizations with higher credits. The higher the value of S_{eval}, the more similar SORep and the basic SO variant become. The value of S_{eval} is supposed to have a substantial impact on savings of model runs.

The question, which can hardly be answered a priori, is how the size of the evaluation stack can be adjusted to find a good compromise between computational savings and reliability of optimized solutions. With respect to reliability, we anticipate that appropriate values of S_{eval} should be of the same magnitude as the number of critical realizations, which are expected to be problem-specific. In order to examine the role of S_{eval} for both efficiency and quality of optimization in this work, we subsequently compare the outcome for different sizes, $S_{\text{eval}} = 5$, 10, 25 and 50.

A lower limit of savings for SORep can be calculated by $(S - S_{\text{eval}})/S$. However, due to occasional break, additional savings can occur. Similar to the approach described for SO and SORed, the OF evaluation starts with the realization that collected the highest credits and "breaks" as soon as a penalized solution is found. The dependency of savings on the selected value of S_{eval} can be seen from the values listed in Table 4. Increasing S_{eval} decreases the savings from about 99% to nearly 94% with similar values for both well cases.

Again, repeated ORs are used to derive a statistical quantification. More accurate results are obtained by 25 ORs instead of only 10 ORs that have been used previously for more computationally demanding implementations. Inspection of the results reveals that the lower S_{eval} is set, the more frequently apparently good solutions of high fitness are identified in the initial search phase of the evolutionary algorithm, but not reproduced later. This is due to the fact that early evaluation is biased by inappropriate criticality of the stack. Post-optimization analysis thus reveals very low values of nominal reliability for these early solutions. Later, a better adaptation of the stack reduces the noise in optimization and enhances an increasingly local search (a feature virtually utilized also in robust GA). As we wish to keep the biased early solutions out of the examination of the results, only the best solutions found during the second half of the search, that is after 43 (single-well case) and 90 (3-well case) generations, are analyzed and discussed here. The results are summarized in Figs. 10 and 11.

Table 4
Median values of optimized pumping rates, reliabilities of optimized solutions and computational savings after 25 ORs for stack ordering procedures with replacement (using evaluation and repository stack)

	Seval	Single well (602 iterations)			3 Wells (1800 iterations)		
		NORep	SORep	SORep decay	NORep	SORep	SORep decay
Pumping rate (l/s)	5	53.5	54.4	43.8	88.4	106.0	88.4
	10	48.2	45.1	31.8	85.7	94.8	63.7
	25	33.1	33.7	32.6	69.8	71.4	61.3
	50	33.2	31.8	32.0	66.4	62.2	61.7
Reliability (%)	5	68.8	73.4	88.9	70.0	40.8	73.9
	10	84.0	90.8	97.8	75.6	70.2	96.2
	25	98.5	99.6	99.8	94.0	94.0	99.2
	50	99.3	100.0	100.0	96.7	99.4	99.4
Savings (%)	5	99.2	99.2	99.5	99.4	99.3	99.4
	10	98.6	98.7	98.9	98.8	98.7	98.9
	25	97.2	97.3	97.5	97.2	97.3	97.9
	50	93.3	94.2	94.4	94.9	94.9	95.1

Savings are calculated by comparing the total number of model runs to the number that would be established if all realizations were used.

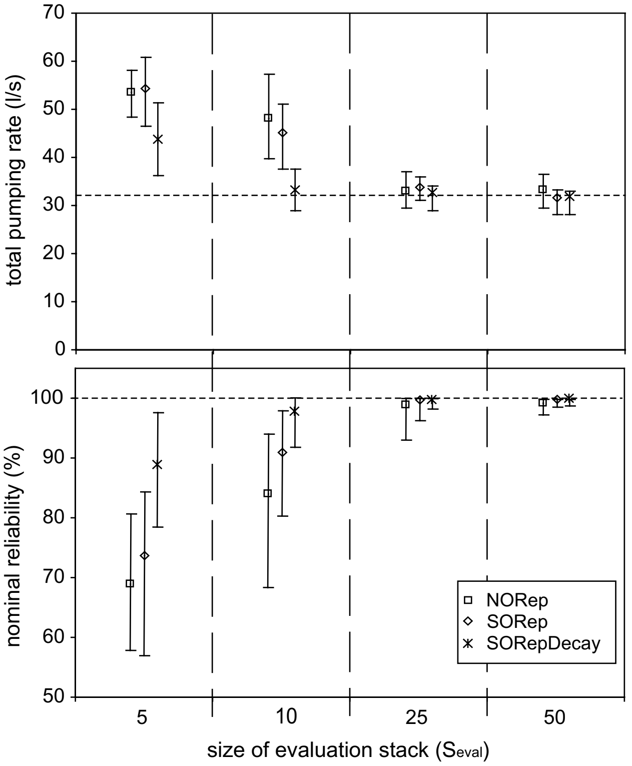

Fig. 10. Single-well problem: median and quantile ranges (10–90%) for implementations using an evaluation stack of different sizes (25 ORs). The dashed line indicates the total pumping rate of the optimized well layout (cp. Table 1).

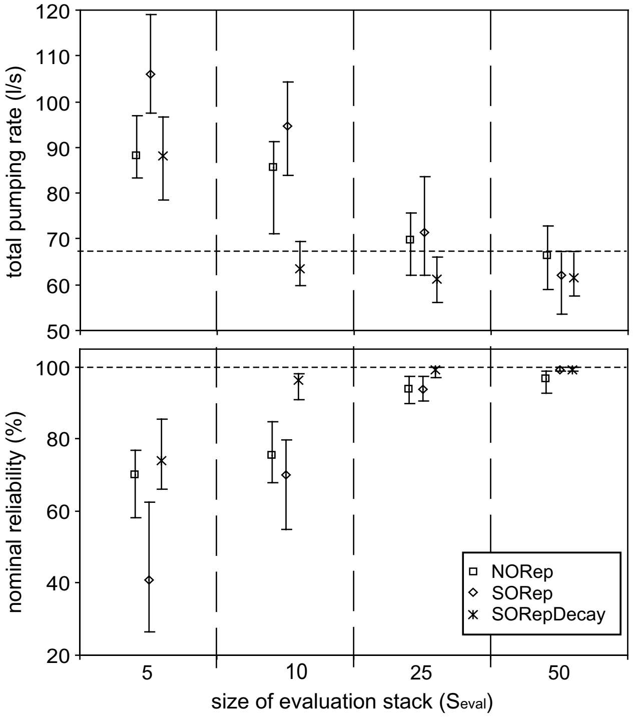

Fig. 11. 3-Well problem: median and quantile ranges (10–90%) for implementations using an evaluation stack of different sizes (25 ORs). The dashed line indicates the total pumping rate of the optimized well layout (cp. Table 1).

When evaluation stack sizes are small, that is $S_{\mathrm{eval}} \leqslant 10$, then both NORep and SORep yield very unsatisfactory results. For both the single- and the 3-well case, the pumping rates are highly overestimated, which yields a low reliability of the candidate solutions. Increasing S_{eval} from 5 to 10 raises the reliability due to the higher number of realizations that determine the objective function value with each iteration. However, the achieved median reliability of 80% is still unsatisfactory. Evidently, both implementations have not considered a sufficient number of critical realizations for iterative evaluation of the objective function. This feature is particularly unexpected for the NORep variant that simulates the noisy GA concept. Calculated reliabilities are lower than the close to 100% values found by [44] for a bioremediation design example. In principle, however, a direct comparison of these results with those obtained in our study is not possible because of the different solvers used. Despite the role of the solver, the performance of the noisy GA seems to depend significantly on the type of problem, its dimension and complexity. In particular, the influence of hydraulic conductivity variance on the fitness landscape appears to be substantial. This is confirmed by the work of Chan Hilton and Culver [12], which shows variable reliabilities of optimized remediation systems that are configured based on a noisy GA. These findings may also explain the comparatively low reliabilities here, where a 3D model of high variance of hydraulic conductivity is utilized.

Further inspection of Fig. 11, namely the achieved reliabilities for $S_{\mathrm{eval}} \leqslant 10$, reveals that NORep outperforms SORep. Here, the optimization evidently does not benefit from focusing on the critical realizations. Reliabilities can even reach values of less than 30%. Simultaneously, total pumping rates are highest. The reason for this can be found in Fig. 12, which shows the (median) number of credited (i.e., potentially critical) realizations over increasing number of iterations. For $S_{\mathrm{eval}} = 5$ and 10, early stagnation is reached at values slightly higher than S_{eval}. These numbers are by far lower than the number of credited realizations computed for stack ordering implementations that exploit all realizations in the evaluation stack (see Tables 2 and 3). In fact, one shortcoming of SORep is an early accumulation of credited realizations in the evaluation stack. As soon as a realization collected a certain number of credits, the restrictive probabilistic sampling procedure implemented here did not allow other realizations to replace existing ones in the evaluation stack, and hence no credits could be allocated to new ones. Consequently, as listed in Table 4 and depicted in Figs. 10 and 11, raising S_{eval} can remarkably enhance nominal reliability of solutions. In both well-cases, the positive effect of high S_{eval} is more pronounced for SORep than for NORep.

6.5. Stack ordering, replacement and decay (SORepDecay)

To overcome the above-mentioned problems introduced by early assembling of the evaluation stack, the sampling could be made more random by reducing the influence of the credits on composition of the evaluation stack (for example: $C^{*} \gg 4$). However, this could also lead to a considerably delayed accumulation of critical realizations, thus resulting in a late convergence. Instead, we recommend incorporating the point in time when credits are assigned, in order to enable a more dynamic sampling of potentially critical realizations. This is based on the assumption that the part of the decision variable space, which is explored during an OR, changes as local search evolves. Realizations that are credited in the initial search phase may not be critical anymore in the later iterations and should therefore be discarded.

Thus, we propose another method, SORepDecay, which denotes the same approach as SORep but with a ‘‘decay’’ factor, k $(0 < k < 1)$. The value of k is multiplied with all existing credits each time when in the previous iteration no penalized solution has been found, that is, no credits have been assigned (Fig. 5). By this decay procedure, a possibly long-term but invalid dominance of credited realizations is mitigated. If the realizations currently forming the evaluation stack do not produce invalid results, that is, they are not critical at this stage of the optimization process, their credits are reduced.

The effect on the number of critical realizations is visualized, in comparison to SORep, in Fig. 12. Independent from S_{eval}, new realizations are continuously assigned credits. As exposed by the small difference between $S_{\mathrm{eval}} = 25$ and 50 for the single-well case, there seems to be an upper limit which closely matches the median value of credited realizations of about 35 reported in Table 1. Differences

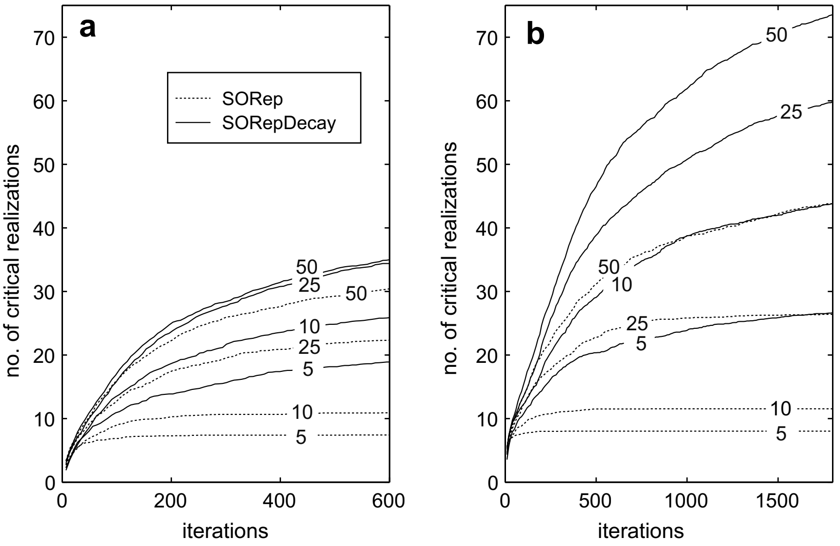

Fig. 12. Median number of credited realizations of 25 ORs for (a) the single-well problem and (b) the 3-well problem. Comparison of two stack ordering methods, SORep and SORepDecay. Numbers refer to the particular size of the evaluation stack.

Please cite this article in press as: Bayer P et al., Computationally efficient stochastic optimization using ..., Adv Water Resour (2007), doi:10.1016/j.advwatres.2007.09.004

are more remarkable for the 3-well case. Increasing S_{eval} always augments the number of credited realizations and, especially for $S_{\mathrm{eval}} = 50$, it seems that this number would increase beyond 75 after more than 1800 iterations.

As demonstrated in Figs. 10 and 11, the most appealing outcomes are produced by SORedDecay. There are substantial discrepancies in the results from SORep and NORep, even for small settings of S_{eval}. For example, for $S_{\mathrm{eval}} = 5$, nearly all single-well solutions achieve nominal reliabilities of more than 80%. For $S_{\mathrm{eval}} = 10$, three solutions already have close-optimal fitness values at nominal reliabilities of 100%. These increase to 12 for $S_{\mathrm{eval}} = 50$. For the problem cases studied here, minor differences exist between the outcomes for $S_{\mathrm{eval}} = 25$ and 50, suggesting that in this case an ideal setting of the evaluation stack size to save a maximum number of model runs is of the order of $S_{\mathrm{eval}} = 25$ and lower.

Of particular interest are the savings in model runs that have been achieved relative to the number of simulations that would have been required without stack ordering (i.e., using NO). We inspected the ORs for $S_{\mathrm{eval}} = 25$ (Table 4) and calculated estimates of between 97.5% (single-well problem) and 98% (three wells). Numerically, this means one OR requires 10–12.5 model runs per iteration on average, instead of 500. Due to the enhanced selection of credited realizations compared to random sampling (NORep) and the restricted accumulation of these realizations in the evaluation stack compared to SORep, computational savings are generally slightly higher. For SORepDecay, the evaluation stack apparently is best up-to-date during the course of the optimization and collects the currently most critical realizations. This enforces early break when successively testing the sampled and ranked realizations.

To apply the most efficient variant, SORepDecay, three parameters have to be set, k, C^* and S_{eval}. Experience from sensitivity analyses with the problem here renders C^* to be of only minor importance. This parameter controls the probability, $p_r = (1 + C_r)/C^*$, of sampling of realizations with low credits, which, in general, have little effect on the performance of the search. An upper limit of C^* should be referred to the maximum credit a realization can get in one iteration which is $(S_{\mathrm{eval}} - 1)^{0.5}$ (Eq. (7)) with S_{eval} representing the size of the evaluation stack. Two aspects have to be considered when specifying S_{eval}. On the one hand S_{eval} should be set as high as possible in order to make sure that all relevant realizations are considered during the optimization. On the other hand high values of S_{eval} are not desirable, because relative savings in computational time are approximately inverse proportional to S_{eval} (Table 4).

Obviously, the quantity of relevant, i.e., potentially critical realizations can not be determined a priori, since criticality is not only an inherent characteristic of a particular realization but also a function of the design of the technical system (here: well field layout) to be evaluated. Note that the layout changes during the course of the optimization when iteratively evaluating candidate solutions.

It is suggested to step-wise augment S_{eval} in successive ORs, starting with a small value, such as $S_{\mathrm{eval}} = 10$, until optimized solutions with sufficiently high reliability are found. In general, the appropriate value of S_{eval} will be affected by the variability of properties of the individual realizations. The lower the variability, the larger the required size of the evaluation stack will be. A small evaluation stack might be sufficient if there are only a small number of realizations that exhibit features causing failure of a broad spectrum of different candidate solutions. By contrast, for the water supply problem considered here, the number of relevant realizations is considerably large. This is due to the fact that particular realizations differ mainly with respect to spatial hydraulic conductivity distribution but only marginally concerning their volumetric water budget (recall that water volume inflow rate varies only ±10% compared to base case). In order to get a rough idea of the quantity of relevant realizations, we analyzed those realizations that represent the binding constraints for optimal single-well systems at fixed positions. Determining the single binding constraint for the optimal solution at each of the 4636 possible well positions separately results in a compilation of more than 50 realizations.

The decay factor, k, can be used to achieve a similar optimization performance at a reduced evaluation stack size. However, the effect of k shrinks for increasing S_{eval} (Figs. 10 and 11, and Table 4). An evident approach, therefore, is to fix k before successively augmenting S_{eval} (see above). The value used in this study, $k = 0.1$, seems to be reasonable. However, a detailed investigation of the joint role of S_{eval} and k will have to be subject to further research.

7. Conclusions and outlook

The new technique of reliability-based stochastic optimization introduced within this study represents a promising extension to existing methods dealing with multiple realizations. This straightforward stack ordering procedure assumes the existence of "worst case" or critical realizations. These represent the binding constraints when a reliability-based objective function is evaluated, regarding numerous model alternatives rather than an exclusively deterministic one. When the models are iteratively called during the optimization, their specific results are compared so as to rank the realizations according to their relevance for the objective function calculation. The experience gained from repeated model calls is utilized to achieve a progressively more mature order in the set of realizations (the stack). This means that the "criticalness" of realizations is not only assessed in relation to one candidate solution which is tested in a single iteration, but to an increasingly representative number. These candidate solutions are search points that delineate the objective function response surface (i.e., the fitness landscape). Accordingly, the iterative learning procedure reveals if critical realizations change among search points. This is particularly essential after the optimization routine has explored the fitness landscape and converged to a (sub-)optimal solution. If, while converging, it is possible to detect those critical realizations, this small set can be used as substitute of the entire stack. This enables significant computational savings for what is (commonly) an extremely demanding optimization with multiple realizations.

An essential element of stack ordering is the use of a noise-insensitive solver that iteratively evaluates the objective function. This is because the highest savings are achieved when approximations of the true objective function values for a stack are obtained from a small sub-set of realizations. The noise produced is shown to be of minor relevance for the performance of a typical stochastic global search algorithm, such as the evolutionary optimization (CMA-ES) method implemented here. The single-objective approach maximizes both fitness and reliability, the latter being crucial for risky water resource management. Nevertheless, as demonstrated for an exemplary fresh water supply problem, hundreds of objective function iterations are still required to adapt the numerous decision variables representing the well layouts of a moving well problem. The high non-linearity and non-convexity of the objective function, which is mainly introduced by the heterogeneity of the hydraulic conductivity, increases the difficulty of detecting an optimal solution. In practice, repeated optimization runs will be necessary to reliably compute the best well configuration.

Based on a demanding hypothetical example, several variants of methods are developed that work with stack ordering. Also, two different well-problems with one and three wells are distinguished. The results of the statistical examination reveal the high potential of each variant tackling multiple realization-based problems that would be virtually untreatable when all realizations are subject to optimization each time the objective function is evaluated. Compared with this, the most efficient variant, stack ordering with replacement and decay (SORepDecay), obtained relative savings in model runs of more than 97.5%. At the same time, the nominal reliability of the optimized solutions was maximized and reached values equal or close to 100%. The SORepDecay variant, which was developed successively after elaborating on the strengths and weaknesses of the procedural elements of the other variants, virtually exhibits only one tuning parameter, which is the size of the evaluation stack S_{eval}. Appropriate values of this parameter for other problems can be expected to vary, although this study indicates that S_{eval} can be relatively small compared with the total stack size.

The transferability of the results to other, different problems of reliability-based optimization can not be predicted by merely considering the findings of this paper. Expectedly, controlling factors will include the type and dimensionality of a problem, as well as stack size and the embodied variability of model parameters. However, solvers such as evolutionary algorithms proved to be suitable for a broad range of problems with different levels of complexity. Alternative solvers are available which may be more efficient in other applications. In particular, the combination of stack ordering with a genetic algorithm is a self-evident alternative to the technique which constituted the focus of this study.

The most appealing feature of the study presented here is the computational efficiency of the procedure, despite the straightforward underlying concept. Although several variant-specific control parameters used have been set empirically, high quality results could be attained. This suggests an enormous potential for stack ordering, which may be used even more efficiently after a more comprehensive parameter analysis. There is also space to amend the variants presented here, for example, by modifications of the probability-based sampling approach, by dynamic adaptation of evaluation stack size or by an alternative implementation of the criteria for filtering critical realizations.

Although of interest to other disciplines of reliability-based design, stack ordering has been particularly developed for stochastic optimization problems in groundwater management, where we face complex objective functions and commonly use multiple equally probable distributed parameter realizations or even multiple conceptual models. The presented implementation is only capable of finding optimal solutions at maximal reliability. Although this reflects an intention of, for example, contaminant groundwater management, variants of the procedure that determine solutions at lower given reliability are conceivable.

A major practical advantage is that the applicability of the presented procedural elements of the stochastic optimization approach is independent of the problem to be solved. There is no insight necessary into the physical processes involved and no (linearity) assumptions to model equations are needed to find a highly reliable optimized solution. At the same time, the automatic identification of critical realizations could be used to identify the main physical controlling parameters by inspecting similarities of those realizations with highest credits.

Acknowledgements

Financial support from the German Research Foundation (Deutsche Forschungsgesellschaft, DFG, Contract No. BA 2850/1-2) for this project is gratefully acknowledged. We thank Anna Bentz and three anonymous reviewers for their comments and suggestions that helped to improve the manuscript.

References

[1] Aly AH, Peralta RC. Optimal design of aquifer cleanup systems under uncertainty using a neural network and a genetic algorithm. Water Resour Res 1999;35(8):2523–32.

[2] Baalousha H, Kongeter J. Stochastic modelling and risk analysis of groundwater pollution using FORM coupled with automatic differentiation. Adv Water Resour 2006;29(12):1815–32.

[3] Baú DA, Mayer AS. Data-worth analysis for multiobjective optimal design of pump-and-treat remediation systems. Adv Water Resour 2007;30(8):1815–30.

[4] Bayer P, Finkel M. Evolutionary algorithms for the optimization of advective control of contaminated aquifer zones. Water Resour Res 2004;40(6). doi:10.1029/2003WR002675.

[5] Bayer P, Finkel M. Optimization of concentration control by CMA-ES: formulation, application and assessment of remedial solutions. Water Resour Res 2007;43. doi:10.1029/2005WR004753.

[6] Bayer P, Finkel M, Teutsch G. Combining pump-and-treat and physical barriers for contaminant plume control. Ground Water 2004;42(6):856–67.

[7] Bierkens MFP. Design a monitoring network for detecting groundwater pollution with stochastic simulation and a cost model. Stoch Environ Res Risk Assess 2006;20:335–51.

[8] Bürger CM, Bayer P, Finkel M. Algorithmic funnel-and-gate system design optimization. Water Resour Res 2007;43:W08426. doi:10.1029/2006WR005058.

[9] Camp CV, Outlaw Jr JE. Stochastic approach to delineating wellhead protection areas. J Water Resour Plan Manage 1998;1244:199–209.

[10] Cantoni M, Marseguerra M, Zio E. Genetic algorithms and Monte Carlo simulation for optimal plant design. Reliab Eng Syst Safety 2000;68(1):29–38.

[11] Carrera J, Alcolea A, Medina A, Hidalgo J, Slooten LJ. Inverse problem in hydrogeology. Hydrogeol J 2005;13(1):206–22.

[12] Chan Hilton AB, Culver TB. Groundwater remediation design under uncertainty using genetic algorithms. J Water Resour Plan Manage 2005;131(1):25–34.

[13] Chan N. Robustness of the multiple realization method for stochastic hydraulic aquifer management. Water Resour Res 1993;29(9): 3159–67.

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

[15] Deutsch CV, Journel AG. GSLIB: Geostatistical Software Library and User's Guide. 2nd ed. New York: Oxford Univ. Press; 1998.

[16] Ditlevsen O, Madsen HO. Structural reliability methods. New York: J. Wiley & Sons; 1996. p. 384.

[17] Feyen L, Gorelick SM. Reliable groundwater management in hydroecologically sensitive areas. Water Resour Res 2004;40(7): W074081–W0740814.

[18] Freeze RA, Massmann H, Smith L, Sperling T, James B. Hydrogeological decision analysis: 1. A framework. Ground Water 1990;28(5):738–66.

[19] Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Reading (MA): Addison-Wesley; 1989.

[20] Gomez-Hernandez JJ, Carrera J. Using linear approximations to rank realizations in groundwater modeling: application to worst case selection. Water Resour Res 1994;30(7):2065–72.

[21] Gopalakrishnan G, Minsker BS, Goldberg DE. Optimal sampling in a noisy genetic algorithm for risk-based remediation design. J Hydroinformat 2003:11–25.

[22] Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. Evol Comput 2001;9(2):159–95.

[23] Hansen N, Müller SD, Koumoutsakos P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 2003;11(1):1–18.

[24] Harbaugh A, Banta E, Hill M, McDonald M. MODFLOW-2000, in the U.S. Geological Survey modular ground-water flow process, U.S Geol. Surv. Open File Rep. 00-92; 2000.

[25] Hendricks Franssen HJWM, Gómez-Hernández JJ. Impact of measurement errors in stochastic inverse conditional modelling by the self-calibrating approach. Adv Water Resour 2003;26(5):501–11.

[26] Jia Y, Culver TB. Robust optimization for total maximum daily load allocations. Water Resour Res 2006;42(2).

[27] Jyrkama MI, Sykes JF. Sensitivity and uncertainty analysis of the recharge boundary condition. Water Resour Res 2006;42(1):W01404. doi:10.1029/2005WR004408.

[28] Kollat JB, Reed PM. Comparing state-of-the-art evolutionary multiobjective algorithms for long-term groundwater monitoring design. Adv Water Resour 2006;29(6):792–807.

[29] Kübert M, Finkel M. Contaminant mass discharge estimation in groundwater based on multi-level point measurements: a numerical evaluation of expected errors. J Contam Hydrol 2006;84(1–2):55–80.

[30] Kunstmann H, Kinzelbach W, Siegfried T. Conditional first-order second-moment method and its application to the quantification of uncertainty in groundwater modelling. Water Resour Res 2002; 38(4):61–615.

[31] Kupfersberger H, Deutsch CV. Ranking stochastic realizations for improved aquifer response uncertainty assessment. J Hydrol 1999; 223(1–2):54–65.

[32] Mantoglou A, Kourakos G. Optimal groundwater remediation under uncertainty using multi-objective optimisation. Water Resour Manage 2007;21(5):835–47.

[33] Marseguerra M, Zio E. Optimizing maintenance and repair policies via a combination of genetic algorithms and Monte Carlo simulation. Reliab Eng Syst Safety 2000;68(1):69–83.

[34] Mayer AS, Kelley T, Miller CT. Optimal design for problems involving flow and transport phenomena in subsurface systems. Adv Water Resour 2002;25:1233–56.

[35] Mehl S, Hill MC. A comparison of solute-transport solution techniques and their effect on sensitivity analysis and inverse modeling results. Ground Water 2001;39(2):300–7.

[36] Minsker BS, Shoemaker CA. Quantifying the effects of uncertainty on optimal groundwater bioremediation policies. Water Resour Res 1998;34(12):3615–25.

[37] Morgan DR, Eheart JW, Valocchi AJ. Aquifer remediation design under uncertainty using a new chance constrained programming technique. Water Resour Res 1993;29(3):551–61.

[38] Mulvey JN, Vanderbrei RJ, Zenios SA. Robust optimization of large scale systems. Oper Res 1995;43(2):264–80.

[39] Pollock DW. User's Guide for MODPATH/MODPATH-PLOT, Vers. 3: A particle tracking post-processing package for MODFLOW, the U.S. Geological Survey finite-difference ground-water flow model, 94-464; 1994.

[40] Ranjithan S, Eheart JW, Garrett Jr JH. Neural network-based screening for groundwater reclamation under uncertainty. Water Resour Res 1993;29(3):563–74.

[41] Reed P, Minsker B, Goldberg DE. Designing a competent simple genetic algorithm for search and optimisation. Water Resour Res 2000;36(12):3757–61.

[42] Saitou K, Izui K, Nishiwaki S, Papalambros P. A survey of structural optimization in mechanical product development. J Comput Informat Sci Eng 2005;5(3):214–26.

[43] Sawyer CS, Lin Y-F. Mixed-integer chance-constrained models for ground-water remediation. J Water Resour Plan Manage 1998;124(5): 285–94.

[44] Smalley JB, Minsker BS, Goldberg DE. Risk-based in situ bioremediation design using a noisy genetic algorithm. Water Resour Res 2000;36(10):3043–52.

[45] Stauffer F, Guadagnini A, Butler A, Franssen H-JH, van de Wiel N, Bakr M, et al. Delineation of source protection zones using statistical methods. Water Resour Manage 2005;19(2):163–85.

[46] Takyi AK, Lence BJ. Surface water quality management using a multiple-realization chance constraint method. Water Resour Res 1999;35(5):1657–70.

[47] Wagner BJ, Gorelick SM. Reliable aquifer remediation in the presence of spatially variable hydraulic conductivity: from data to design. Water Resour Res 1989;25(10):2211–25.

[48] Wu J, Zheng C, Chien CC, Zheng L. A comparative study of Monte Carlo simple genetic algorithm and noisy genetic algorithm for cost-effective sampling network design under uncertainty. Adv Water Resour 2006;29(6):899–911.

[49] Zheng C, Wang PP. An integrated global and local optimization approach for remediation system design. Water Resour Res 1999; 35(1):137–48.