

Optimal Well Placement Under Uncertainty Using a Retrospective Optimization Framework

Honggang Wang, SPE, David Echeverría Ciaurri, SPE, and Louis J. Durlofsky, SPE, Stanford University, and Alberto Cominelli, SPE, Eni

Summary

Subsurface geology is highly uncertain, and it is necessary to account for this uncertainty when optimizing the location of new wells. This can be accomplished by evaluating reservoir performance for a particular well configuration over multiple realizations of the reservoir and then optimizing based, for example, on expected net present value (NPV) or expected cumulative oil production. A direct procedure for such an optimization would entail the simulation of all realizations at each iteration of the optimization algorithm. This could be prohibitively expensive when it is necessary to use a large number of realizations to capture geological uncertainty. In this work, we apply a procedure that is new within the context of reservoir management—retrospective optimization (RO)—to address this problem. RO solves a sequence of optimization subproblems that contain increasing numbers of realizations. We introduce the use of k -means clustering for selecting these realizations. Three example cases are presented that demonstrate the performance of the RO procedure. These examples use particle swarm optimization (PSO) and simplex linear interpolation (SLI)-based line search as the core optimizers (the RO framework can be used with any underlying optimization algorithm, either stochastic or deterministic). In the first example, we achieve essentially the same optimum using RO as we do using a direct optimization approach, but RO requires an order of magnitude fewer simulations. The results demonstrate the advantages of cluster-based sampling over random sampling for the examples considered. Taken in total, our findings indicate that RO using cluster sampling represents a promising approach for optimizing well locations under geological uncertainty.

Introduction

In recent years, the application of computational optimization for reservoir management has become more commonplace. Optimization algorithms can be applied, for example, to maximize NPV or cumulative oil recovered. These procedures can be used to optimize the time-varying control of existing wells (e.g., determine the optimal flow rates or bottomhole pressures as a function of time) or the location and type of new wells. Optimization is in general expensive computationally, because many flow simulations are typically required.

The efficient treatment of uncertainty, particularly geological uncertainty, represents a key outstanding challenge in the practical application of these procedures. This is because, in order to represent the high degree of uncertainty in the reservoir geology, many geological realizations must be considered. Then, optimizations can be performed by maximizing the expected value of the objective function over all of the realizations available. Direct application of such an approach, in which all realizations are considered at every iteration of the optimization, will lead to very costly optimizations, particularly when the number of realizations is large.

The goal of this work is to introduce and apply an efficient procedure, RO, for optimizing the placement of new wells over

multiple geological models. The key feature of RO is that it does not treat all realizations at all iterations of the optimization algorithm. Rather, RO defines a sequence of approximate optimization subproblems, which sequentially account for increasing numbers of geological realizations. We show that, by applying a k -means clustering procedure to define the realizations included in each subproblem, the performance of the RO algorithm can be improved. RO was developed within the context of operations research (Chen and Schmeiser 2001; Wang and Schmeiser 2008) and has not been applied previously for reservoir-management problems. The approach can be used with any core (deterministic or stochastic) optimization algorithm.

For practical problems with intractably large numbers of realizations, we can still apply the general RO approach by using increasing numbers of realizations at each RO iteration and terminating the algorithm without simulating all of the realizations. Depending on the uncertainties being considered and the data available, the selection of subsets of realizations could be based on subsurface properties (e.g., depositional-system type, water-oil and gas-oil contacts), fluid parameters, and rock/fluid parameters. Approaches based on experimental design may be very useful in some cases. In the examples presented here, we apply k -means clustering, and the attributes used account for variability in subsurface properties.

There have been a number of previous studies addressing well-placement optimization and optimization under geological uncertainty. A variety of different algorithms have been applied for the well-placement optimization problem. These include stochastic search algorithms such as genetic algorithms (Artus et al. 2006; Güyagüler and Horne 2004; Yeten et al. 2003), particle swarm optimization (Onwunalu and Durlofsky 2010), and simultaneous perturbation stochastic approximation (Bangerth et al. 2006). Deterministic approaches have also been applied (Sarma and Chen 2008; Zandvliet et al. 2008). Stochastic approaches have the advantage of searching globally, which can be important given the often significantly nonsmooth objective functions associated with the well-placement problem, though they typically require many more function evaluations than gradient-based approaches.

Optimization of reservoir performance under uncertainty has also been considered by a number of researchers. This includes studies addressing both the well-placement problem (Artus et al. 2006; Güyagüler and Horne 2004; Onwunalu and Durlofsky 2010; Özdoğan and Horne 2006; Yeten et al. 2003) and the well-control problem (Aitokhuehi and Durlofsky 2005; Chen et al. 2009; van Essen et al. 2009; Wang et al. 2009). In these applications, optimization was performed over a fixed number of realizations (in many cases, relatively few realizations), which, as indicated previously, will be very expensive if many realizations are considered. The RO technique applied here differs from previous approaches in that a sequence of optimization problems is solved using increasing numbers of geological models.

This paper is organized as follows. We first present the problem of well placement under uncertainty. We then provide a detailed description of the RO framework. The core optimization algorithms applied in this work—particle swarm optimization and simplex-linear-interpolation-based gradient search—are then discussed. Next, three examples that demonstrate the effectiveness of the RO procedure are presented. These examples include

TABLE 1—ALGORITHM 1: RETROSPECTIVE OPTIMIZATION

```

1: for  $k=1,2,\dots$  do
2:   Sample  $N_k$  realizations  $\xi_k=\{\omega_1,\dots,\omega_{N_k}\}$ 
3:    $\mathbf{x}_k = \text{optSolver}(\mathbf{x}_{k-1}, \xi_k, \mathbb{X})$ 
4: end for
    
```

the Brugge model and two synthetic cases involving 80 and 405 realizations. The results clearly demonstrate the computational advantages of the RO procedure. We conclude with a summary and recommendations.

RO for Optimizing Well Placement

Optimization Problem. We seek to optimize the positions of a number of wells under geological uncertainty. This uncertainty is expressed in terms of a collection of model realizations. A general formulation for optimizing under uncertainty is as follows. Find a solution \mathbf{x}^* in \mathbb{X} that maximizes

$$J = E_{\Omega}[G(\mathbf{x}, \omega)], \dots\dots\dots (1)$$

where E_{Ω} represents the expectation function over the set of all realizations Ω and G is a numerical process that calculates the sample observation of the cost function J for a given \mathbf{x} and the realization ω . In our case, \mathbf{x} is the vector defining the locations of the wells and $G(\mathbf{x}, \omega)$ is evaluated by performing a flow simulation with the geological model defined by realization ω and the well locations defined by \mathbf{x} . Note that the well locations are the same in all realizations.

Within the context of oil-reservoir problems, the set Ω is finite. If we denote the number of realizations available by N , then $\Omega = \{\omega_i\}_{i=1}^N$, and the expected value operator can be written as $E_{\Omega}[G(\mathbf{x}, \omega)] = \sum_{i=1}^N G(\mathbf{x}, \omega_i) p(\omega_i)$, where $p(\omega_i)$ is the probability associated with realization ω_i . Here, we will consider N equally probable realizations [i.e., $p(\omega_i) = 1/N$], though this is not a requirement of the method.

The cost functions considered in this work are the expected NPV and the expected cumulative oil produced. The decision variables are positive integers (i.e., $\mathbb{X} \subset \mathbb{N}^n$) that specify the locations of gridblocks containing vertical wells. In the 3D example, the decision variables also include the initial and final layers in which the well is completed. Therefore, the set \mathbb{X} is bounded, and the problem has a finite number of feasible solutions. We note that deviated or multilateral wells can also be treated within the RO framework, and that we could additionally include well-control variables in the optimizations.

RO Framework. RO (Chen and Schmeiser 2001; Wang and Schmeiser 2008) solves a sequence of sample-path optimization problems where the true objective function in Eq. 1 is approximated by the average over the selected samples/realizations. The number of realizations (sample size) is increased from subproblem to subproblem, and the initial solution for the current subproblem is simply the returned solution from the previous subproblem. In a population-based algorithm such as a genetic algorithm or particle-swarm optimization, the initial generation/swarm for the current subproblem is simply the last generation/swarm from the previous subproblem. In early iterations, the RO algorithm does not require excessive computations because the sample sizes are small. In late iterations, when evaluation of the (approximate) objective function is expensive because many realizations are considered, the initial solutions are typically closer to the optimum. Thus, fewer iterations of the core optimizer are required. RO can therefore be expected to achieve computational savings relative to a direct optimization that considers all N realizations at every iteration.

An RO algorithm (Table 1; Algorithm 1) using a general core optimizer, referred to as optSolver, is outlined in the following. Given a set of realizations (provided by the RO algorithm) at RO

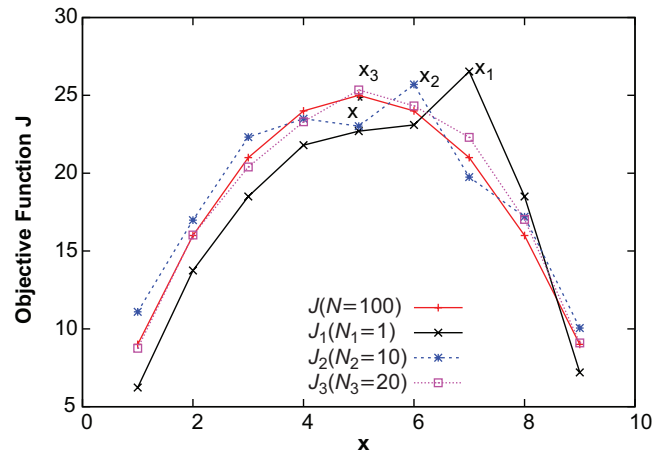


Fig. 1—RO for a simple function with one optimization variable.

iteration k , optSolver determines the optimal well locations based on maximization of expected value.

Given: a bounded feasible region $\mathbb{X} \subset \mathbb{N}^n$, an initial solution $x_0 \in \mathbb{X}$, a simulation evaluation process G , an increasing sequence of sets of realizations $\{\xi_k\}_{k=1,2,\dots}$ where $\xi_k = \{\omega_1, \dots, \omega_{N_k}\}$.

Return: A maximizer \mathbf{x}^* of J .

The RO framework generates a sequence of sample-path problems $P_k, k = 1, 2, \dots$. In the k th sample-path iteration, optSolver provides an optimal solution (or a sufficiently close approximation of the optimal solution) \mathbf{x}_k of $E_{\xi_k}[G(\mathbf{x}, \omega)] = \sum_{i=1}^{N_k} G(\mathbf{x}, \omega_i) / N_k$. Note that, in each subproblem, we take $p(\omega_i) = 1/N_k$. As k increases, the sample-path problem P_k and solution \mathbf{x}_k approach the true problem and the maximizer \mathbf{x}^* of J .

We illustrate in Fig. 1 the application of the RO framework to a simple example with one optimization variable. Each realization is given by $G(x, \omega_i) = x(10 - x) + \omega_i$, where ω_i is random noise chosen such that $J(x) = (1/N) \sum_{i=1}^N G(x, \omega_i) = x(10 - x)$. We take $N = 100$ realizations, and \mathbb{X} is the set of the first eleven non-negative integers. In this case, one evaluation of J at any integer $x \in [0, 10]$ involves 100 function evaluations. Instead of maximizing $J(x)$ over 100 function evaluations each time, RO considers a sequence of subproblems with 1, 10, 20, and all 100 realizations (the sequence of sample sizes is selected heuristically, as discussed in the following). The first optimization problem P_1 is inexpensive because computing every objective function requires only one function evaluation. As we can see in Fig. 1, in this example the first optimal solution x_1 is relatively close to the true optimum x^* . Next, RO solves P_2 with $N_2 = 10$ realizations sampled from the total set of 100 realizations. With x_1 as the initial solution for P_2 , RO returns the solution x_2 , which is very near the true optimum x^* . The third sample-path problem P_3 involves $N_3 = 20$ realizations, and it yields the solution x_3 . The fourth sample-path problem, which considers all 100 realizations, converges in a single iteration because the initial guess (x_3) is equal to the true optimum x^* .

We stress that, in this simple case of optimization under uncertainty, we are able to find the solution for a problem with 100 realizations based on the solutions from intermediate problems with 1, 10, and 20 realizations. As we will see later, efficiency will also be achieved for well-placement problems (i.e., many fewer function evaluations are performed than would be required if all the realizations were considered at all times). We note, finally, that certain algorithmic parameters, such as the number of realizations to use in each sample-path problem and the stopping criteria associated with the optSolver as a function of P_k (there is no need to solve the early sample-path problems to full convergence), clearly impact algorithm performance.

Core Optimization Algorithms. As indicated previously, RO can be used with any core optimization algorithm optSolver. In this work, PSO and SLI gradient search are used. We now provide general descriptions of these two optimization techniques.

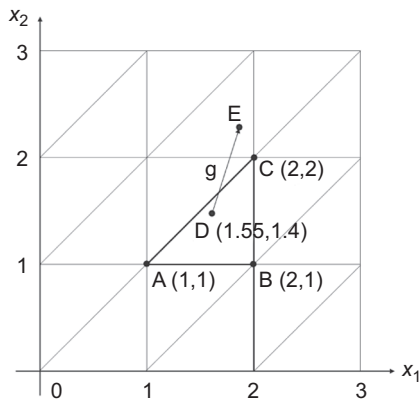


Fig. 2—Simplex interpolation in a 2D space. The point D is enclosed by the simplex with vertices A, B, and C.

PSO. PSO is a population-based stochastic optimization procedure originally developed by Kennedy and Eberhardt (1995) and Eberhardt and Kennedy (1995). The method has since been used in many application areas and has recently been applied for well-placement optimization (Onwunali and Durlofsky 2010, 2011). These references should be consulted for full algorithmic details.

In the PSO algorithm, a point in the search space (i.e., a possible solution) is called a particle, and the collection of particles at a given iteration is referred to as the swarm. At each iteration, all particles move to new positions in the search space. Let $\mathbf{x}^{k,i} \in \mathbb{R}^n$ be the position of Particle i at Iteration k . Let $\mathbf{y}^{k,i}$ represent the best position (solution) found by Particle i up to Iteration k , and let $\mathbf{y}_{x_i}^{*,k}$ be the best position found by any of the particles in the neighborhood of \mathbf{x}^i up to Iteration k . The neighborhood structure defines the particles that Particle i “sees,” and a variety of neighborhoods have been suggested for PSO. In this work, the neighborhood includes all other particles in the swarm; thus, our algorithm is referred to as a “global-best” PSO. In this case, $\mathbf{y}_{x_i}^{*,k}$ is the best solution visited by any particle in the swarm up to Iteration k .

The new position of Particle i at Iteration $k + 1$, $\mathbf{x}^{k+1,i}$, is computed by adding a velocity, $\mathbf{v}^{k+1,i} \in \mathbb{R}^n$, to the current position $\mathbf{x}^{k,i}$:

$$\mathbf{x}^{k+1,i} = \mathbf{x}^{k,i} + \mathbf{v}^{k+1,i}.$$

The velocity $\mathbf{v}^{k+1,i}$ is computed as follows:

$$\mathbf{v}^{k+1,i} = \alpha \mathbf{v}^{k,i} + c_1 \mathbf{D}_1 (\mathbf{y}^{k,i} - \mathbf{x}^{k,i}) + c_2 \mathbf{D}_2 (\mathbf{y}_{x_i}^{*,k} - \mathbf{x}^{k,i}), \dots \dots \dots (2)$$

where α , c_1 , and c_2 are prescribed weights and \mathbf{D}_1 and \mathbf{D}_2 are diagonal matrices with each component sampled randomly from the interval (0,1). The three contributions to particle velocity are referred to as the inertia, cognitive, and social components (Eberhardt and Kennedy 1995). The inertia component $\mathbf{v}^{k,i}$ causes the particle to continue in the direction in which it was moving at iteration k . The cognitive term (involving c_1) captures the particle memory regarding its previous best position, and provides a velocity component in this direction. The social component (term involving c_2) represents information about the best position of any particle (in this implementation) and causes movement toward that particle. The velocity at Iteration $k + 1$ represents a combination of these three components. Thus, each particle moves to a new position based on its existing trajectory, its own memory, and the collective experience of other particles. The overall algorithm parallelizes naturally because the flow simulations required for each particle can be readily performed in a distributed manner.

SLI-Based Gradient Search. SLI (Wang and Schmeiser 2008; Weiser and Zarantonello 1988) uses a continuous relaxation of the cost function J based on function evaluations at integer points. In this paper, we apply only linear interpolation because it is

computationally less expensive and easier to implement than higher-order approximations. In addition, the original discrete-valued optimizers of J are also real-valued optimizers for the relaxed cost function. Here, we designate this relaxed cost function by \bar{J} .

SLI uses $n + 1$ integer points (simplex) to define a piecewise-linear surface in \mathbb{R}^n . The gradient obtained from an SLI [sometimes called a simplex gradient; see Kelley (1999)] is the same for all of the points contained in the interpolation simplex. Given any continuous solution \mathbf{x} , SLI first identifies the $n + 1$ integer vertices of a simplex that encloses \mathbf{x} . The interpolation function $\bar{J}(\mathbf{x})$ is a linear combination of $J(\mathbf{x}_i)$, with coefficients determined from the distances of \mathbf{x} to the simplex vertices $\{\mathbf{x}_i\}_{i=1}^{n+1}$. Refer to Weiser and Zarantonello (1988) and Wang and Schmeiser (2008) for details on SLI. In Fig. 2, we illustrate SLI for an objective function J that varies over a 2D search space. In this case, to compute $\bar{J}(D)$, SLI finds the enclosing simplex with vertices A, B, and C. The linear interpolation for \bar{J} at the continuous Point D can then be written based on the location of D relative to the vertices of this simplex. Using the values in Fig. 2, we have $\bar{J}(D) = 0.45J(A) + 0.15J(B) + 0.4J(C)$.

The SLI-based gradient search implemented for \bar{J} in this study is a steepest-ascent procedure with inexact line search. In other words, SLI-based gradient search finds an (approximate) locally optimal solution by performing a sequence of line searches. SLI generates a piecewise-linear function \bar{J} for an integer valued J and computes the gradient of \bar{J} for any interior continuous point $\mathbf{x} \notin \mathbb{X}$ in the convex hull of \mathbb{X} . If $\mathbf{x} \in \mathbb{X}$, we simply select a vector from the subgradient (Rockafellar 1970) of \bar{J} at \mathbf{x} . For each line search, given a starting solution \mathbf{x}_0 , SLI first calculates the function value $\bar{J}(\mathbf{x}_0)$ and the gradient $\nabla \bar{J}(\mathbf{x}_0)$. A standard line-search routine (Rockafellar 1970) from \mathbf{x}_0 in the direction of $\nabla \bar{J}(\mathbf{x}_0)$ basically solves the following 1D optimization problem:

$$\max_{s>0} \bar{J}(x_0 + s \nabla \bar{J}(x_0)), \dots \dots \dots (3)$$

where the scalar variable s is the step size. In our numerical implementation, SLI searches for an approximate optimal step size \hat{s} by sequentially applying step sizes within the geometric sequence $\{0.5, 1, 2, 4, 8, \dots\}$ until $\bar{J}(\mathbf{x}_0 + \hat{s} \nabla \bar{J}(\mathbf{x}_0))$ yields no additional improvement. If none of the step sizes in the sequence provides an increase in the cost function, the SLI search terminates and returns the best integer solution evaluated over the iterations (note that the SLI search always evaluates at least the $n + 1$ integer points that constitute the initial simplex). Though SLI search does not use an exact line-search routine, it has been observed to perform efficiently for the examples considered in this work.

It should be stressed that the $n + 1$ simulations related to the simplex enclosing \mathbf{x} can be computed in parallel. The trial points within a line search require only the cost-function evaluation and not the associated gradient. For these function evaluations, interpolation must still be performed, but in this case \bar{J} can be approximated by considering only the nearby vertices (i.e., using fewer than $n + 1$ integer points). This reduces the number of serial flow simulations that must be performed.

Clustering Within RO. The particular realizations included in the subset ξ_k for the subproblem P_k can significantly impact the performance of the RO approach. As discussed earlier, depending on the available data and estimated parameter ranges, various sampling strategies can be applied. In this work, we apply a k -means clustering in MATLAB (Seber 1984) that, as we will see, generally leads to more-efficient optimizations than the use of a random-sampling procedure. Note that, in the random-sampling procedure, no systematic approach for sampling is applied. In this case N_k realizations are simply selected randomly from the full set.

In k -means clustering, we first establish a mapping from each realization to a finite (and normally very small) number m of attributes. Examples of attributes that can be useful in optimal well placement are cumulative field oil production, original oil in place (OOIP), or some measure of permeability. In our implementation, each of these quantities is first normalized to [0,1]. Then, for

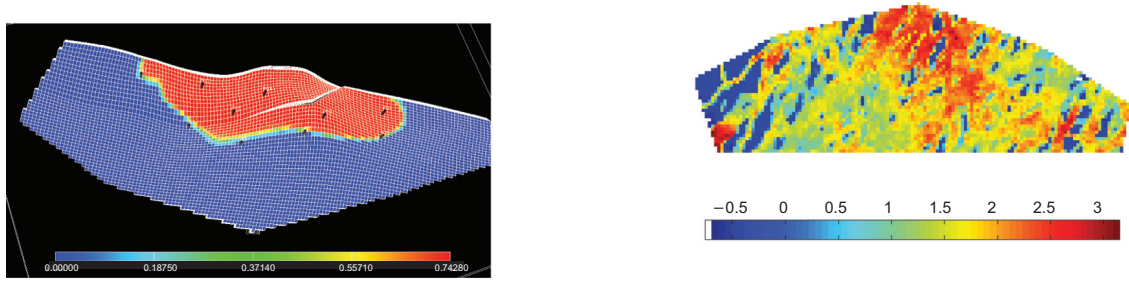


Fig. 3—Initial oil saturation and permeability (\log_{10} scale) for first layer of first realization of the Brugge model.

quantities that vary in time (cumulative oil production) or space (permeability), a vector of values is associated with each realization. Using this approach, we are able to identify each realization ω_i with a vector of attributes $\mathbf{p}_i \in \mathbb{R}^m$. These vectors are entered into the k -means algorithm.

Application of k -means clustering then provides k vectors in \mathbb{R}^m (cluster centers) that minimize the average distance d for all N realizations, given as follows:

$$d = \sum_{i=1}^N \min_{j=1, \dots, k} \|\mathbf{p}_i - \boldsymbol{\pi}_j\|^2, \dots \dots \dots (4)$$

where $\boldsymbol{\pi}_j$ denotes the coordinates of the center of Cluster j . The optimization associated with Eq. 4 does not involve expensive function evaluations, so the computational cost of the clustering will be negligible when compared to the main simulation-based optimization process. Once we have the cluster centers, we can attach a particular realization ω_i to one of the centers by simply computing $\arg \min_{j=1, \dots, k} \|\mathbf{p}_i - \boldsymbol{\pi}_j\|$. When the clustering involves a quantity that varies in time, such as cumulative oil production, we compute distances based on the L_1 norm, which approximates the area between the curves.

If the clustering involves only attributes based on static parameters, such as OOIP or average permeability, then the clustering need be performed only once. If the clustering also involves attributes that depend on the well configuration, such as cumulative oil production, then the clusters must be recomputed over the course of the optimization. Because this computation involves simulation of all realizations, it will be expensive if it is performed frequently. In this work, we regenerate N_k clusters at each iteration of the RO algorithm (i.e., once for each subproblem P_k), except at the last RO iteration, where all realizations are used and no clustering is necessary. One realization is then sampled randomly from each of the N_k clusters. In computing the approximate objective $E_{\xi_k}[G(\mathbf{x}, \omega)]$, we assign the probability for each realization simulated to be $1/N_k$. This weighting could be varied to reflect the number of realizations in each cluster (i.e., models drawn from clusters containing many realizations would be weighted higher), though this approach was not investigated here.

In the examples here, we use four or five RO iterations. The computations associated with the clustering for these cases represent approximately a 10% additional cost. We note that it may be beneficial to regenerate the clusters during an RO iteration, though this was not investigated here.

The selection of the sequence of sample sizes used in this work relies on heuristics. Our approach here is to use 1 to 5 realizations for the first one or two subproblems; around ten, or a few tens, of realizations for the next two or three subproblems; and all of the realizations (which for the problems considered here range from 80 to 405) for the last subproblem. In limited tests, this choice did not seem to strongly impact the effectiveness of the RO approach, though further research to formalize (or optimize) this strategy will be useful.

Optimization Results Using RO Framework

In this section, we assess the performance of the RO procedure for several example problems. We consider three different models and two different core optimizers. Both random sampling and cluster sampling are applied. We note that our goal in this work

is not to identify the best core-optimization algorithm (this choice will depend on the particular problem under study), but rather to demonstrate that RO is suitable for use with both stochastic and deterministic procedures. In general, stochastic techniques such as PSO are designed for global search with potentially rough cost functions, while local methods such as SLI are more efficient but do not search globally. We use the ECLIPSE™ reservoir simulator for the first two cases and the streamline simulator 3DSL for the third case.

Brugge Case. The first example involves a slight modification of the 3D Brugge synthetic model (Peters et al. 2010). Uncertainty is quantified by means of 104 permeability and porosity realizations originally provided in the Brugge model. In this work, the water-oil contact for these realizations is taken as a random variable with uniform distribution in [1668 m, 1688 m]. The model for each realization contains $139 \times 48 \times 9$ (total of 60,048) gridblocks, with each block of dimensions $100 \times 100 \times 6$ m. Unlike the original Brugge model, this reservoir has five existing (fully penetrating) vertical water-injection wells. For this example, we use PSO as the core optimizer and seek to determine the locations and initial and final completion layers of five new production wells. The total production time is 30 years. Injection wells are prescribed to operate at 180 bar, and production wells at 50 bar. Fig. 3 shows initial oil saturation and permeability for one realization.

The objective function J is the expected total NPV for 30 years of production over the $N = 104$ realizations of the reservoir model. The objective function is thus defined as

$$J = \frac{1}{N} \sum_{j=1}^N \left[\sum_{i=1}^{30} \frac{p_o q_{i,j}^o - p_{w_1} q_{i,j}^{w_1} - p_{w_2} q_{i,j}^{w_2}}{(1 + \lambda)^i} \right], \dots \dots \dots (5)$$

where p_o is oil price (USD 80/bbl); p_{w_1} and p_{w_2} are the water-production and -injection costs (both taken to be USD 5/bbl), respectively; $\lambda = 0.0234$ is the yearly discount rate; and $q_{i,j}^o$, $q_{i,j}^{w_1}$, and $q_{i,j}^{w_2}$ are the (yearly) oil- and water-production rates and the water-injection rate, respectively, for the i th year and the j th model realization. These rates are all in units of bbl/yr. One call of G gives one observation of the production quantities $q_{i,j}^o, q_{i,j}^{w_1}, q_{i,j}^{w_2}, i = 1, 2, \dots, 30$, for a given \mathbf{x} and ω_j , which in turn provides one observation of J . It takes approximately 3 minutes to perform a single simulation run using a 2.66-GHz CPU and 1.95 GB of RAM.

For this example, we consider random sampling, cluster sampling, and a direct or “full” (with all realizations) optimization. In the full optimization (referred to as PSO-Full), RO is not applied. Rather, at each iteration of the optimization, we evaluate the performance of all 104 realizations and then apply Eq. 5. The swarm size for this run is 20 and the number of PSO iterations is 100. This is very expensive, but it does provide a reference case for evaluating the RO procedures. In the RO runs, we use five sample-path problems. The sequence of sample sizes $\{1, 5, 16, 21, 104\}$ is used for sample-path problems $P_k, k = 1, \dots, 5$, respectively. The last sample-path problem is the true problem. This sequence was chosen heuristically based on the number of realizations. In the core PSO procedure, we use a swarm size of 20 and proceed for 20, 15, 10, 5, and 1 iterations for each of the five subproblems, respectively. As noted earlier, the initial swarm for Subproblem P_k is the last swarm from Subproblem P_{k-1} .

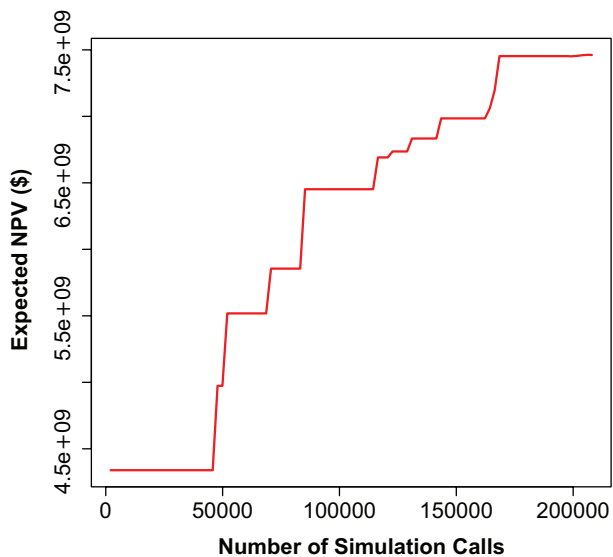


Fig. 4—Performance of direct optimization (PSO-Full) for the Brugge case.

The performance of PSO-Full is shown in Fig. 4. It is evident that, using this procedure, approximately 200,000 simulations are required to find the optimal solution and that the expected NPV is approximately USD 7.46×10^9 . We next consider the performance of RO-PSO with random sampling. The progress of the optimization for this case is shown in Fig. 5. The vertical jumps between the different curves correspond to iterations in the RO procedure (i.e., moving from P_k to P_{k+1}). At later RO iterations, there are also noticeable horizontal gaps between the curves. These occur because a single PSO iteration requires a large number of simulation calls when a substantial number of realizations is considered. From this figure, we see that the optimal solution, corresponding to an expected NPV of approximately USD 7.2×10^9 , is achieved after approximately 10,000 simulations. Thus, this optimization required approximately a factor of 20 times fewer function evaluations than PSO-Full. Later, we will present results for multiple PSO runs, which are useful in allowing us to draw conclusions regarding the relative performance of the two approaches.

We next consider the application of RO-PSO with cluster sampling. The settings for PSO are as in the case with random sampling. The clustering is performed based on normalized

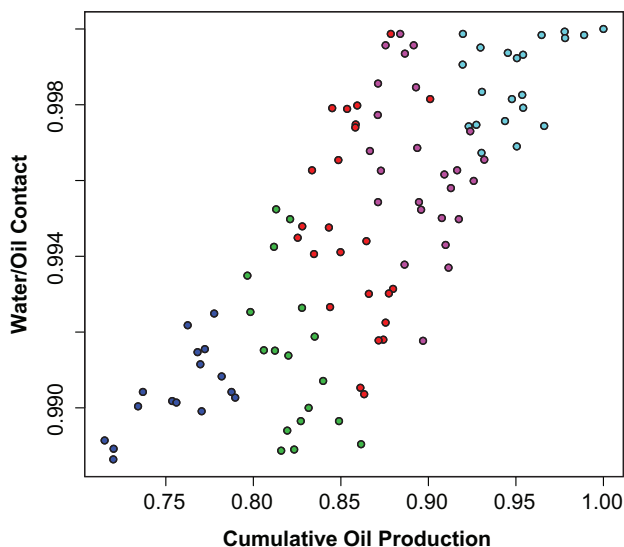


Fig. 6—Five clusters for the 104 realizations of the Brugge case.

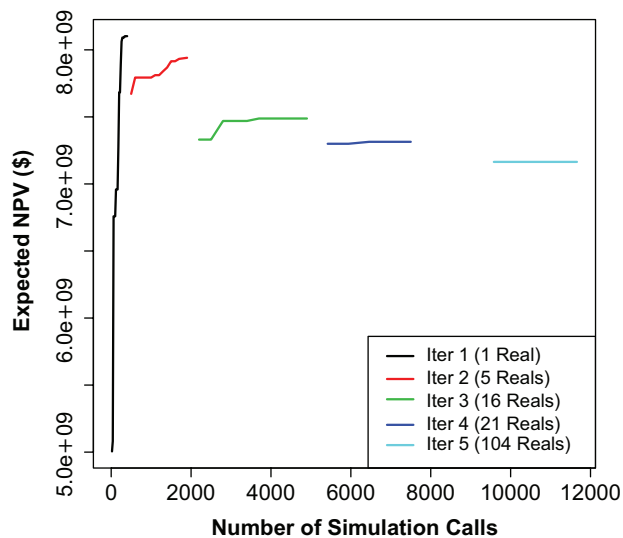


Fig. 5—Performance of random sampling RO-PSO for the Brugge case.

cumulative oil production, OOIP, and location of the water-oil contact. Fig. 6 shows the five clusters obtained from the k -means clustering method (here we present the projection into two dimensions, so only the two most important attributes are plotted). The performance of RO-PSO is shown in Fig. 7. The progress of the optimization is different from that using random sampling (Fig. 5), and we see that the optimal expected NPV is approximately USD 7.6×10^9 . This value is larger than that achieved using either PSO-Full or RO-PSO with random sampling.

Because PSO is a stochastic optimizer, it is useful to run it multiple times to draw conclusions regarding algorithm performance. For this reason, both random- and cluster-sampling RO-PSO were run two more times (the PSO-Full procedure is very expensive, so additional runs were not performed). Of the most interest is the expected NPV of the current solution evaluated over all 104 realizations (i.e., even though the optimizer does not consider all 104 realizations until the last sample-path problem P_5 , the quantity of interest is still expected NPV over all realizations). This quantity, averaged over the three runs, is presented for the two RO methods in Fig. 8. The maximum expected NPV (averaged over the three runs) for RO-PSO with cluster sampling is USD 7.61×10^9 , while

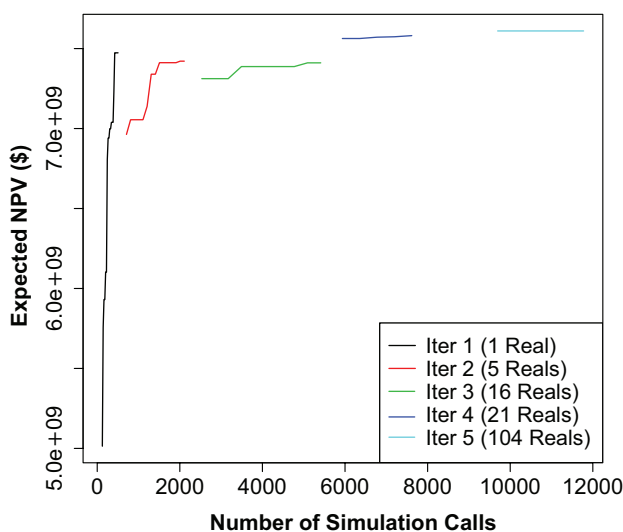


Fig. 7—Performance of cluster-sampling RO-PSO for the Brugge case.

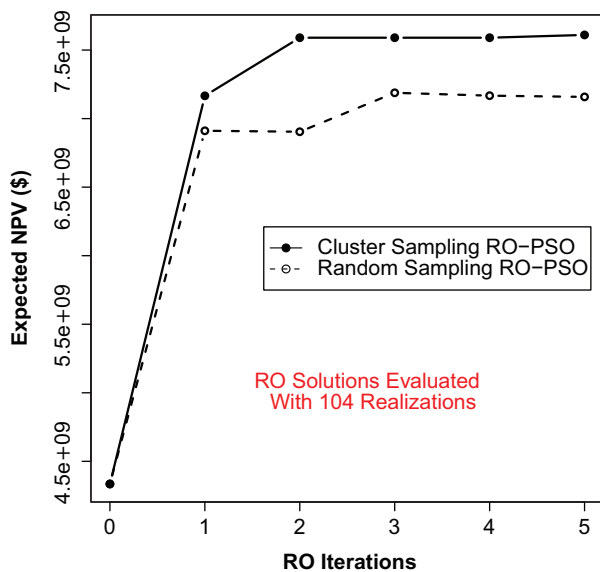


Fig. 8—Performance of RO-PSO with random and cluster sampling for the Brugge case (average of three runs).

that for RO-PSO with random sampling (achieved at the third RO iteration) is USD 7.45×10^9 . These values are to be compared with that for the PSO-Full procedure (USD 7.46×10^9). For a given number of reservoir simulations, we expect RO-PSO with cluster sampling to outperform the other two approaches, as is observed. We reiterate, however, that more runs of all algorithms would be required to draw broader conclusions.

In Fig. 8, we see that the cluster-sampling method, on average, essentially achieves its optimum after two RO iterations, while the random-sampling method requires at least three RO iterations (and variation continues even up to P_5). This observation is significant; later RO iterations can be much more expensive than earlier RO iterations because larger numbers of realizations are considered as the algorithm progresses. This also suggests that, in practice, we may be able to terminate RO-PSO with cluster sampling without considering all realizations. Note that, if we did not evaluate the current solution over all 104 realizations for RO-PSO with random sampling, we would take as the optimized solution the result at the last RO iteration, which is USD 7.42×10^9 . This illustrates the fact that the RO algorithm is not guaranteed to provide an increase in objective function with RO iteration because the optimization problem changes at each RO iteration. This is not observed, however, for RO-PSO with cluster sampling, presumably because the clustering preserves key problem features from one RO iteration to the next.

In Fig. 9, we show the initial and optimized well locations from one optimization run using RO-PSO with cluster sampling. The

allowable region is indicated by the rectangle. The five production wells have moved from their random initial locations toward the boundary. These shifts result in a 75% increase in the expected NPV for this case.

The results in this section clearly demonstrate the advantages of the RO procedure relative to the “full” PSO approach and, within the RO procedure, the advantage of cluster sampling relative to random sampling.

Channelized 2D Example. The second example involves 80 realizations of a 2D synthetic channelized model. Each realization contains 100×64 gridblocks, with each block of dimensions $100 \times 100 \times 50$ m. Four permeability realizations are shown in Fig. 10. The field is produced through waterflooding. The goal of the optimization is to determine the location of six new vertical production wells to maximize the expected cumulative oil produced over 50 years of production. Injection wells operate at 34.5 bar and production wells at 15.2 bar. This case is of interest because we use SLI-based gradient search as the core optimizer rather than PSO. We note that PSO, or a hybrid procedure, could also have been applied, but our intent here is to demonstrate the use of RO with a local optimizer.

For this example, we set the RO parameters as follows. We use four sample-path problems with the sample sizes set to $\{4, 8, 24, 80\}$. The last subproblem with 80 realizations represents the original problem. We again compare random sampling and cluster sampling. For cluster sampling, the clusters are based on normalized cumulative field oil production, permeability distance, and OOIP. Permeability distance d_i is defined as the Euclidian distance between the permeability field of Realization i , designated \mathbf{m}_i , and the average permeability field over all 80 realizations, designated $\bar{\mathbf{m}}$ (i.e., $d_i = \|\mathbf{m}_i - \bar{\mathbf{m}}\|_2$).

In Fig. 11, cumulative-oil-production curves for all 80 realizations, using the initial well locations, are plotted. Fig. 12 shows eight clusters of realizations identified by applying k -means clustering. The cumulative oil production associated with the cluster centroids is shown in Fig. 13. We see that the range of performance for the 80 realizations in Fig. 11 is captured in Fig. 13.

The performance of one RO-SLI run using random sampling and cluster sampling is shown in Figs. 14 and 15, respectively. For this case, the difference between the two approaches is significant—we observe much larger jumps from subproblem to subproblem in the random-sampling results.

We performed five additional optimizations using RO-SLI with both random and cluster sampling for this case. Different initial guesses were used for the various runs, which resulted in different (local) optimal solutions. Averaged results for the expected cumulative production of the current solution (evaluated over all 80 realizations) for the six runs are shown in Fig. 16. Here again, consistent with the RO-PSO results shown in Fig. 8, we see that cluster sampling provides better average performance than random sampling.

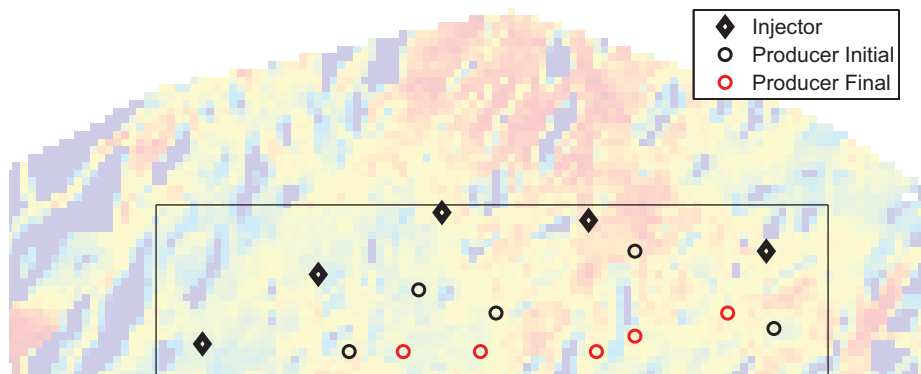


Fig. 9—Optimized well locations using RO-PSO with cluster sampling for the Brugge case.

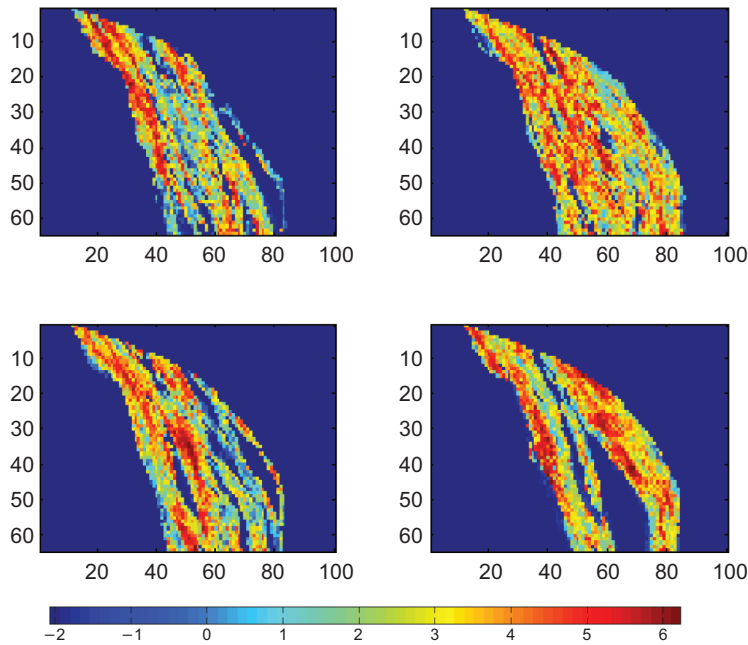


Fig. 10—Four permeability realizations of the channelized model.

In Fig. 17, we show the initial and final well locations for one run of cluster-sampling RO-SLI. For this optimization run, the final well locations are close to the initial solution, which is not surprising because SLI is a local optimizer. The expected cumulative oil production increases by approximately 60% in this case.

Example With Multiple Training Images. Our final example involves a large number of realizations (namely 405). These realizations, taken from Scheidt and Caers (2008), were derived from 81 different training images (five realizations were generated from each training image). Each realization contains 80×80 gridblocks. Six of the 405 realizations are shown in Fig. 18. For this case, we seek to determine the locations of six vertical wells (three injectors and three producers) to maximize expected cumulative oil produced. The injection and production wells all operate at a rate of 2,000 barrels per day.

We apply only RO-SLI with cluster sampling in this example. The initial clusters, shown in Fig. 19, are based on normalized cumulative oil production (using the initial well configuration), permeability distance, and OOIP (the plot does not show the OOIP axis). We use five sample-path problems with the sequence of sample sizes {4, 8, 16, 32, 405}. Fig. 20 shows the progress of RO-SLI

with cluster sampling for this problem. Results for the expected cumulative-oil production of the current solution evaluated over all 405 realizations are shown in Fig. 21. From this figure, we see that the RO procedure has essentially converged after only approximately 5,000 simulations. This is fast convergence, given the large number of realizations considered in this example. This result highlights the suitability of cluster-based RO for problems involving large numbers of geological realizations.

Conclusions

In this paper, we introduced a new approach (within the context of reservoir management), RO, for optimizing the locations of new wells under geological uncertainty. The RO framework is compatible with any underlying optimization algorithm. Here, we considered both stochastic (PSO) and deterministic (SLI-based gradient search) core optimizers. RO optimizes using a sequence of realizations, and two approaches were considered for determining these realizations: a random sampling and a cluster sampling.

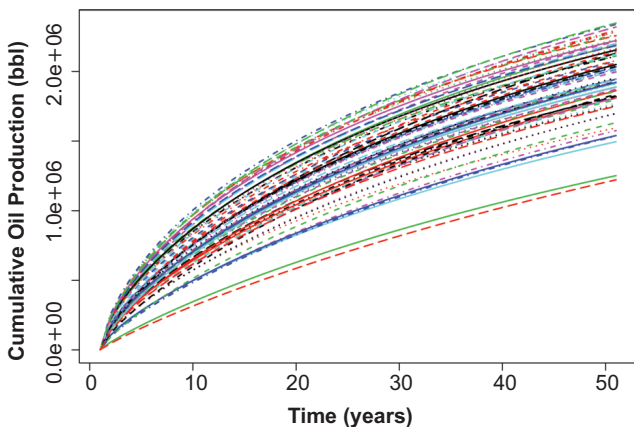


Fig. 11—Cumulative oil production for all 80 realizations for the channelized example.

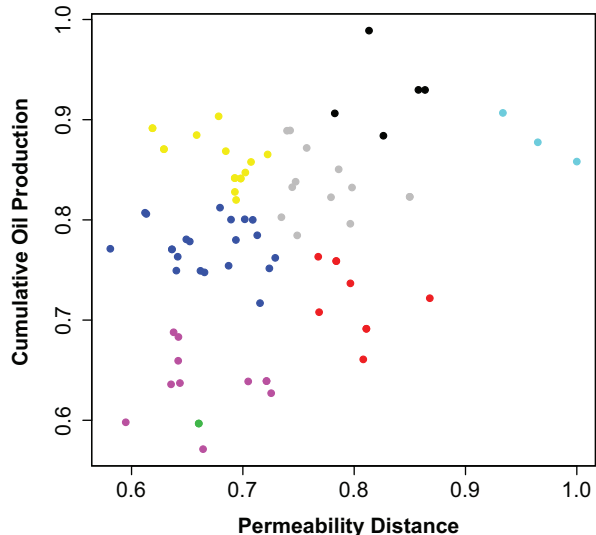


Fig. 12—Eight clusters of the 80 realizations for the channelized example.

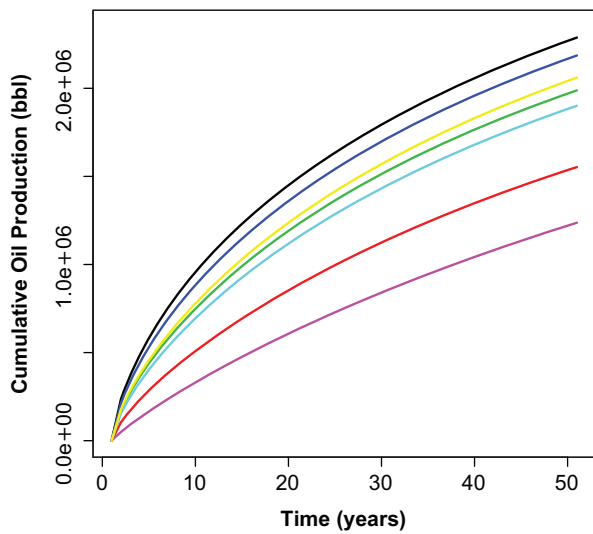


Fig. 13—Cumulative oil production for the eight cluster centroids for the channelized example.

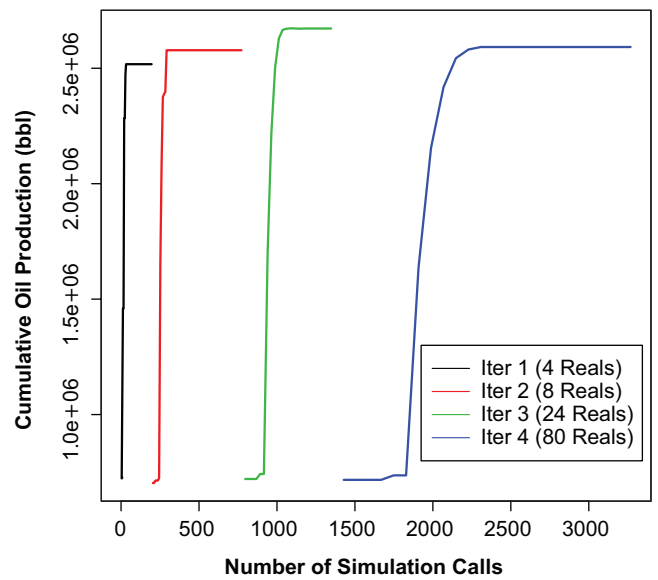


Fig. 14—Performance of random-sampling RO-SLI for the channelized example.

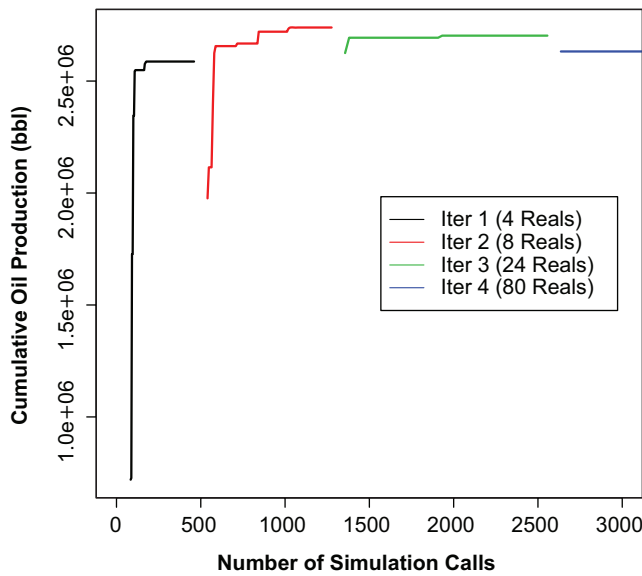


Fig. 15—Performance of cluster-sampling RO-SLI for the channelized example.

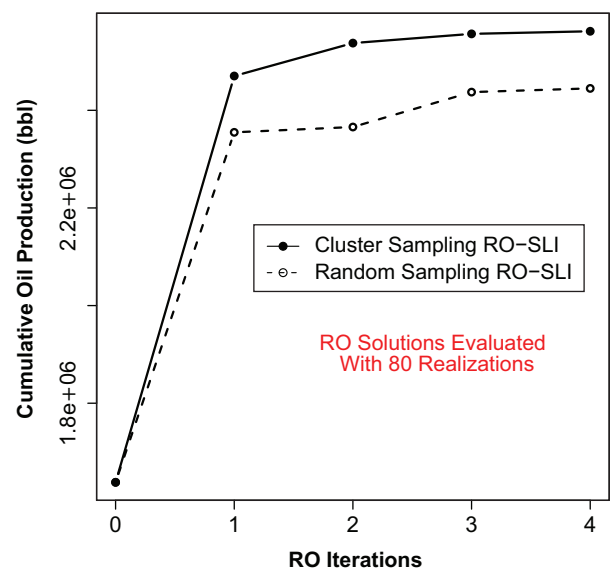


Fig. 16—Performance of RO-SLI with random and cluster sampling for the channelized example (average of six runs).

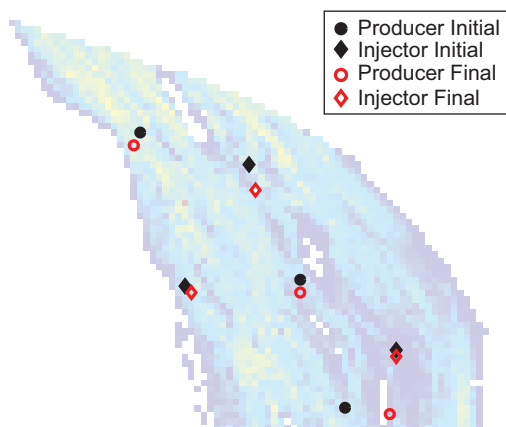


Fig. 17—Optimized well locations using RO-SLI with cluster sampling for the channelized example.

In the cluster sampling, a *k*-means clustering based on a few attributes is applied to select realizations. The clustering must be updated periodically during the optimization if one or more of the attributes depend on the well configuration.

Results for the first example (Brugge case), which used PSO as the core optimizer, demonstrated that RO requires significantly fewer reservoir-simulation runs than a direct or “full” optimization that considers all realizations at each iteration. We also showed that, for both the Brugge case and a channelized example, cluster-based sampling within the RO procedure appears preferable to random sampling. In the third case, 405 realizations from 81 different training images were considered. For this case, no comparisons were performed, but the RO-SLI algorithm with cluster sampling was able to find a (essentially) converged solution after only approximately 5,000 simulations.

Taken in total, the findings presented here demonstrate that the RO framework is well suited for use in optimization under geological uncertainty. Future work should be directed toward further testing of the overall optimization framework, determina-

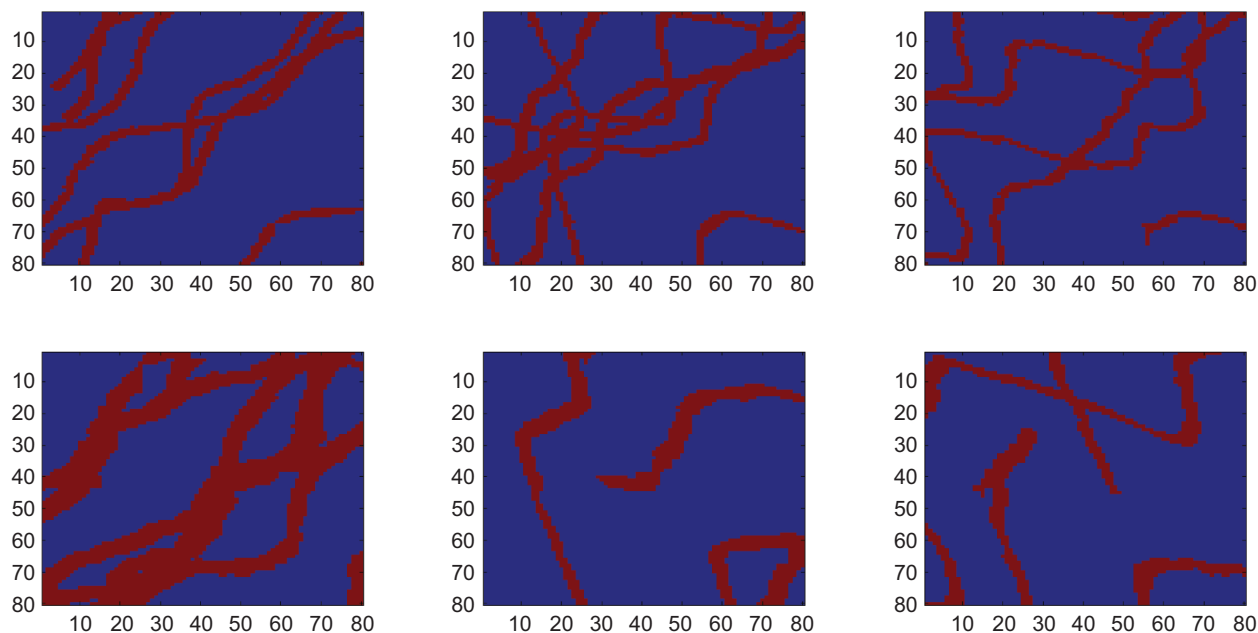


Fig. 18—Six of the 405 permeability realizations for the example with multiple training images (red is high-permeability sand; blue is low-permeability background).

tion of the appropriate sequence of sample sizes to be used in RO, establishment of guidelines for the frequency of updating the clusters, and determination of the weights to be used in computing the approximate objective function. The use of experimental design for selecting realizations should additionally be considered. It will also be of interest to apply the procedure for multiobjective optimization and for combined well-placement and well-control optimization.

Nomenclature

- G = sample observation of the cost function
- J = objective function
- \bar{J} = simplex-linear-interpolation function
- N = total number of realizations
- N_k = number of realizations for k th subproblem
- P_k = k th retrospective-optimization subproblem

- \mathbf{v} = PSO velocity function
- \mathbf{x} = well locations
- \mathbf{x}_0 = initial well locations
- \mathbf{x}^* = optimal well locations
- \mathbb{X} = set of feasible solutions
- ξ_k = subset of realizations in k th subproblem
- ω = particular realization
- Ω = population of reservoir realizations

Acknowledgments

We are grateful to Eni SpA. for financial support. We thank Andrea Luigi Lamberti and Dario Rametta (Eni) for discussions and assistance with some of the simulations and Celine Scheidt (Stanford University) for providing the models used in the last example.

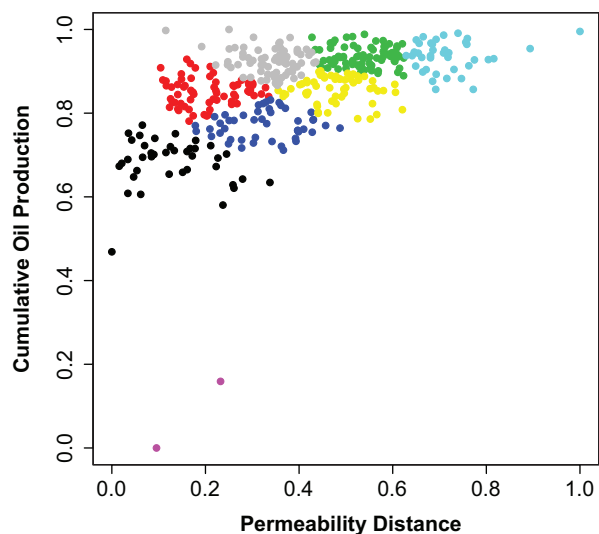


Fig. 19—Eight clusters of the 405 realizations for the example with multiple training images.

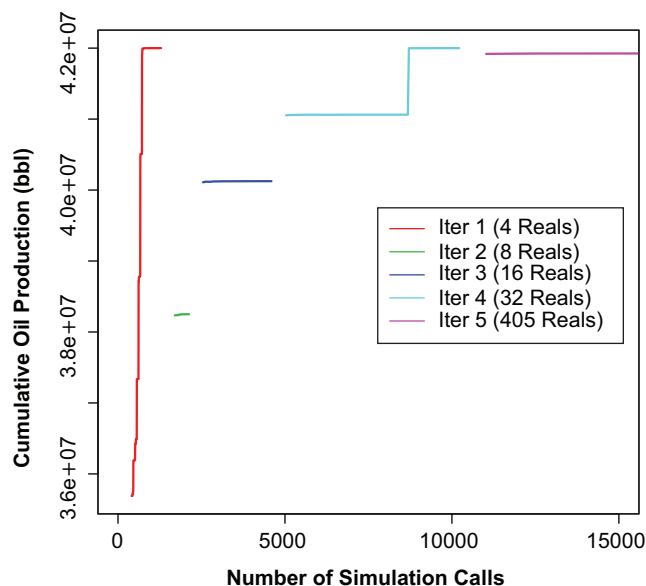


Fig. 20—Performance of cluster sampling RO-SLI for the example with multiple training images.

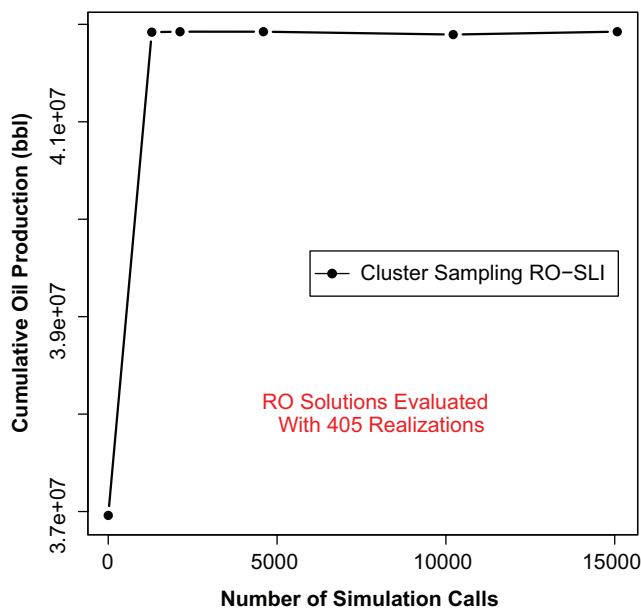


Fig. 21—RO-SLI with cluster sampling for the example with multiple training images.

References

- Aitokhuehi, I. and Durlofsky, L.J. 2005. Optimizing the performance of smart wells in complex reservoirs using continuously updated geological models. *J. Pet. Sci. Eng.* **48** (3–4): 254–264. <http://dx.doi.org/10.1016/j.petrol.2005.06.004>.
- Artus, V., Durlofsky, L.J., Onwunali, J.E., and Aziz, K. 2006. Optimization of nonconventional wells under uncertainty using statistical proxies. *Comput. Geosci.* **10** (4): 389–404. <http://dx.doi.org/10.1007/s10596-006-9031-9>.
- Bangerth, W., Klie, H., Wheeler, M.F., Stoffa, P.L., and Sen, M.K. 2006. On optimization algorithms for the reservoir oil well placement problem. *Comput. Geosci.* **10** (3): 303–319. <http://dx.doi.org/10.1007/s10596-006-9025-7>.
- Chen, H. and Schmeiser, B.W. 2001. Stochastic root finding via retrospective approximation. *IIE Trans.* **33** (3): 259–275. <http://dx.doi.org/10.1080/07408170108936827>.
- Chen, Y., Oliver, D.S., and Zhang, D. 2009. Efficient Ensemble-Based Closed-Loop Production Optimization. *SPE J.* **14** (4): 634–645. SPE-112873-PA. <http://dx.doi.org/10.2118/112873-PA>.
- Eberhardt, R.C. and Kennedy, J. 1995. A new optimizer using particle swarm theory. *Proc., Sixth International Symposium on Micro Machine and Human Science (MHS '95)*, Nagoya, Japan, 4–6 October, 39–43. <http://dx.doi.org/10.1109/MHS.1995.494215>.
- Güygüler, B. and Horne, R.N. 2004. Uncertainty Assessment of Well-Placement Optimization. *SPE Res Eval & Eng* **7** (1): 24–32. SPE 87663. <http://dx.doi.org/10.2118/87663-PA>.
- Kelley, C.T. 1999. *Iterative Methods for Optimization*. Philadelphia, Pennsylvania: Frontiers in Applied Mathematics, SIAM.
- Kennedy, J. and Eberhardt, R.C. 1995. Particle swarm optimization. *Proc., 1995 IEEE International Conference on Neural Networks*, Perth, Western Australia, 27 November–1 December, 1942–1947.
- Onwunali, J.E. and Durlofsky, L.J. 2010. Application of a particle swarm optimization algorithm for determining optimum well location and type. *Comput. Geosci.* **14** (1): 183–198. <http://dx.doi.org/10.1007/s10596-009-9142-1>.
- Onwunali, J.E. and Durlofsky, L.J. 2011. A New Well-Pattern-Optimization Procedure for Large-Scale Field Development. *SPE J.* **16** (3): 594–607. SPE-124364-PA. <http://dx.doi.org/10.2118/124364-PA>.
- Özdoğan, U. and Horne, R.N. 2006. Optimization of Well Placement Under Time-Dependent Uncertainty. *SPE Res Eval & Eng* **9** (2): 135–145. SPE-90091-PA. <http://dx.doi.org/10.2118/90091-PA>.
- Peters, L., Arts, R.J., Brouwer, G.K., et al. 2010. Results of the Brugge Benchmark Study for Flooding Optimization and History Matching. *SPE Res Eval & Eng* **13** (3): 391–405. SPE-119094-PA. <http://dx.doi.org/10.2118/119094-PA>.
- Rockafellar, R.T. 1970. *Convex Analysis*. Princeton, New Jersey: Princeton Landmarks in Mathematics, Princeton University Press.
- Sarma, P. and Chen, W.H. 2008. Efficient Well Placement Optimization with Gradient-based Algorithms and Adjoint Models. Paper SPE 112257 presented at the Intelligent Energy Conference and Exhibition, Amsterdam, 25–27 February. <http://dx.doi.org/10.2118/112257-MS>.
- Scheidt, C. and Caers, J. 2008. Representing Spatial Uncertainty Using Distances and Kernels. *Math. Geosci.* **41** (4): 397–419. <http://dx.doi.org/10.1007/s11004-008-9186-0>.
- Seber, G.A.F. 1984. *Multivariate Observations*. New York: Wiley Series in Probability and Statistics, John Wiley & Sons.
- van Essen, G.M., Zandvliet, M.J., van den Hof, P.M.J., Bosgra, O.H., and Jansen, J.D. 2009. Robust Waterflooding Optimization of Multiple Geological Scenarios. *SPE J.* **14** (1): 202–210. SPE-102913-PA. <http://dx.doi.org/10.2118/102913-PA>.
- Wang, C., Li, G., and Reynolds, A.C. 2009. Production Optimization in Closed-Loop Reservoir Management. *SPE J.* **14** (3): 506–523. SPE-109805-PA. <http://dx.doi.org/10.2118/109805-PA>.
- Wang, H.G. and Schmeiser, B.W. 2008. Discrete stochastic optimization using linear interpolation. In *Proc. 2008 Winter Simulation Conference*, ed. S.J. Mason, R.R. Hill, L. Mönch, O. Rose, T. Jefferson, and J.W. Fowler, 502–508. Piscataway, New Jersey: IEEE Press.
- Weiser, A. and Zarantonello, S.E. 1988. A note on piecewise linear and multilinear table interpolation in many dimensions. *Math. Comput.* **50** (181): 189–196. <http://dx.doi.org/10.1090/S0025-5718-1988-0917826-0>.
- Yeten, B., Durlofsky, L.J., and Aziz, K. 2003. Optimization of Nonconventional Well Type, Location, and Trajectory. *SPE J.* **8** (3): 200–210. SPE-86880-PA. <http://dx.doi.org/10.2118/86880-PA>.
- Zandvliet, M.J., Handels, M., Van Essen, G.M., Brouwer, D.R., and Jansen, J.D. 2008. Adjoint-Based Well-Placement Optimization Under Production Constraints. *SPE J.* **13** (4): 392–399. SPE-105797-PA. <http://dx.doi.org/10.2118/105797-PA>.

Honggang Wang is an assistant professor in the Department of Industrial and Systems Engineering at Rutgers University. He has finished his 2-year postdoctoral research in energy resources engineering at Stanford University. His research interests lie in stochastic system modeling and optimization, and their applications in energy-production systems. Wang holds a BS degree in power engineering from Shanghai Jiao Tong University and a PhD degree in industrial engineering from Purdue University.

David Echeverría Ciaurri is with the Department of Petroleum and Energy Analytics at the IBM Thomas J. Watson Research Center. He has worked at the Dutch National Research Institute for Mathematics and Computer Science, at the Lawrence Livermore National Laboratory, and at Stanford University. He holds a degree in telecommunication engineering and an MS degree in applied mathematics from the University of Zaragoza, Spain, as well as a PhD degree in numerical analysis from the University of Amsterdam.

Louis J. Durlofsky is the Otto N. Miller Professor and Chair in the Department of Energy Resources Engineering at Stanford University. He codirects the Stanford University Industrial Affiliate Program on Reservoir Simulation (SUPRI-B) and the Stanford Smart Fields Consortium. He was previously affiliated with Chevron Energy Technology Company. Durlofsky holds a BS degree from Penn State, as well as MS and PhD degrees from the Massachusetts Institute of Technology, all in chemical engineering.

Alberto Cominelli is the Reservoir Modelling Technical Advisor for Eni's E&P Division. His interests cover all aspects of reservoir modeling process, including computational methods for reservoir simulation, upscaling, history matching, 4D seismic, and integration between surface and subsurface modeling. Cominelli holds an Msc degree in physics from the University of Pavia, Italy.