Comput Geosci (2006) 10: 303–319
DOI 10.1007/s10596-006-9025-7

On optimization algorithms for the reservoir oil well placement problem

W. Bangerth · H. Klie · M. F. Wheeler · P. L. Stoffa · M. K. Sen

Received: 6 November 2004 / Accepted: 24 April 2006 / Published online: 17 August 2006

Abstract Determining optimal locations and operation parameters for wells in oil and gas reservoirs has a potentially high economic impact. Finding these optima depends on a complex combination of geological, petrophysical, flow regimen, and economical parameters that are hard to grasp intuitively. On the other hand, automatic approaches have in the past been hampered by the overwhelming computational cost of running thousands of potential cases using reservoir simulators, given that each of these runs can take on the order of hours. Therefore, the key issue to such automatic optimization is the development of algorithms that find good solutions with a minimum number of function evaluations. In this work, we compare and analyze the efficiency, effectiveness, and reliability of several optimization algorithms for the well placement problem. In particular, we consider the simultaneous perturbation stochastic approximation (SPSA), finite difference gradient (FDG), and very fast simulated annealing (VFSA) algorithms. None of these algorithms guarantees to find the optimal solution, but we show that both SPSA and VFSA are very efficient in finding nearly optimal solutions with a high probability. We illustrate this with a set of numerical experiments based on real data for single and multiple well placement problems.

Keywords reservoir optimization · reservoir simulation · simulated annealing · SPSA · stochastic optimization · VFSA · well placement

W. Bangerth (✉)
Department of Mathematics, Texas A&M University, College Station, TX 77843, USA
e-mail: bangerth@math.tamu.edu

W. Bangerth
Institute for Geophysics, The University of Texas at Austin, Austin, TX 78712, USA

H. Klie · M. F. Wheeler
Center for Subsurface Modeling, Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin, TX 78712, USA

H. Klie
e-mail: klie@ices.utexas.edu

M. F. Wheeler
e-mail: mfw@ices.utexas.edu

P. L. Stoffa · M. K. Sen
Institute for Geophysics, John A. and Katherine G. Jackson School of Geosciences, University of Texas at Austin, Austin, TX 78712, USA

P. L. Stoffa
e-mail: pauls@utig.ig.utexas.edu

M. K. Sen
e-mail: mrinal@utig.ig.utexas.edu

1. Introduction

The placement, operation scheduling, and optimization of one or many wells during a given period of the reservoir production life has been a focus of attention of oil production companies and environmental agencies in the last years. In the petroleum engineering scenario, the basic premise of this problem is to achieve the maximum revenue from oil and gas while minimizing operating costs, subject to different geological and economical constraints. This is a challenging problem because it requires many large-scale reservoir

simulations and needs to take into account uncertainty in the reservoir description. Even if a large number of scenarios is considered and analyzed by experienced engineers, the procedure is in most cases inefficient and delivers solutions far from the optimal one. Consequently, important economical losses may result.

Optimization algorithms provide a systematic way to explore a broader set of scenarios and aim at finding very good or optimal ones for some given conditions. In conjunction with specialists, these algorithms provide a powerful mean to reduce the risk in decision-making. Nevertheless, the major drawback in using optimization algorithms is the cost of repeatedly evaluating different exploitation scenarios by numerical simulation based on a complex set of coupled nonlinear partial differential equations on hundreds of thousands to millions of gridblocks. Therefore, the major challenge in relying on automatic optimization algorithms is finding methods that are efficient and robust in delivering a set of nearly optimal solutions.

In the past, a number of algorithms have been devised and analyzed for optimization and inverse problems in reservoir simulation (see, e.g., [26, 34, 49]). These problems basically fall within four categories: (1) history matching; (2) well location; (3) production scheduling; (4) surface facility design. In the particular case of the well placement problem, the use of optimization algorithms began to be reported about 10 years ago (see, e.g., [4, 37]). Since then, increasingly complicated cases have been reported in the literature, mainly in the directions of complex well models (type, number, orientation), characteristics of the reservoir under study, and the numerical approaches employed for its simulation [5, 13, 14, 30, 46, 47]. From an optimization standpoint, most algorithms employed so far are either stochastic or heuristic approaches; in particular, this includes simulated annealing (SA) [4] and genetic algorithms (GA) [14, 47]. Some of them have also been combined with deterministic approaches to provide a fast convergence close to the solution; for instance, GA with polytope and tabu search [5] and GA with neural networks [7, 14]. In all cases, the authors point out that all these algorithms are still computationally demanding for large-scale applications.

In this work, we introduce the simultaneous perturbation stochastic approximation (SPSA) method to the well placement problem, and compare its properties with other, better-known algorithms. The SPSA algorithm can be viewed as a stochastic version of the steepest descent method, where a stochastic vector replaces the gradient vector computed using point-wise finite difference approximations in each of the directions associated with the decision variables. This generates a highly efficient method because the number of function evaluations per step does not depend on the dimension of the search space. Despite the fact that the algorithm has a local search character in its simplest form, the stochastic components of the algorithm are capable of delivering nearly optimal solutions in relatively few steps. Hence, we show that this approach is more efficient than other traditional algorithms employed for the well placement problem. Moreover, the capability of the SPSA algorithm to adapt easily from local search to global search makes it attractive for future hybrid implementations.

This paper is organized as follows: section 2 reviews the components that are used for the solution of the well placement problem: (1) the reservoir model and the parallel reservoir simulator used for forward modeling; (2) the economic revenue function to optimize; and, (3) the optimization algorithms considered under this comparative study. Section 3 shows an exhaustive and comparative treatment of these algorithms based on the placement of one well. Section 4 extends the previous numerical analysis to the placement of multiple wells. Section 5 summarizes the results of the present work and discusses possible directions of further research.

2. Problem description and approaches

In this study, we pose the following optimization question: At which position should a (set of) new injection/production wells be placed to maximize the economic revenue of production in the future? We first detail the description of the reservoir simulator that allows the evaluation of this economic revenue objective function. We then proceed to describe the objective function and its dependence on coefficients such as costs for injection and production. We end the section by describing the algorithms that were considered for solving this optimization problem.

2.1. Description and solution of the oil reservoir model

For the purpose of this paper, we restrict our analysis to reservoirs that can be described by a two-phase, oil–water model. This model can be formulated as the following set of partial differential equations for the conservation of mass of each phase $m = o, w$ (oil and water):

$$\frac{\partial(\phi N_m)}{\partial t} + \nabla \cdot U_m = q_m. \tag{1}$$

Here, ϕ is the porosity of the porous medium, N_m is the concentration of the phase m, and q_m represents the source/sink term (production/injection wells). The fluxes U_m are defined by using Darcy's law [15], which with gravity ignored – reads as $U_m = -\rho_m K \lambda_m \nabla P_m$, where ρ_m denotes the density of the phase, K the permeability tensor, λ_m the mobility of the phase, and P_m the pressure of phase m. Additional equations specifying volume, capillary, and state constraints are added, and boundary and initial conditions complement the system (see [2, 15]). Finally, $N_m = S_m \rho_m$ with S_m denoting saturation of a phase. The resulting system for this formulation is

$$\frac{\partial(\phi \rho_m S_m)}{\partial t} - \nabla \cdot (\rho_m K \lambda_m \nabla P_m) = q_m. \tag{2}$$

In this paper we consider wells that either produce (a mixture of) oil and water, or at which water is injected. At an injection well, the source term q_w is nonnegative (we use the notation $q_w^+ := q_w$ to make this explicit). At a production well, both q_o and q_w may be nonpositive, and we denote this by $q_m^- := -q_m$. In practice, both injection and production rates are subject to control, and thus to optimization; however, in this paper we assume that rates are indirectly defined through the specification of the bottom hole pressure (BHP). These rates are used for evaluating the objective function and are not decision parameters in our optimization problem.

This model is discretized in space by using the *expanded mixed finite element* method, which, in the case considered in this paper, is numerically equivalent to the *cell-centered finite difference* approach [1, 38]. In time, we use a fully implicit formulation solved at every time step with a Newton–Krylov method preconditioned with algebraic multigrid.

The discrete model is solved with the IPARS (Integrated Parallel Accurate Reservoir Simulator) software developed at the Center for Subsurface Modeling at The University of Texas at Austin (see, e.g., [25, 32, 35, 43, 45]). IPARS is a parallel reservoir simulation framework that allows for different algorithms or formulations (IMPES, fully implicit), different physics (compositional, gas–oil–water, air–water, and one-phase flow) and different discretizations in different parts of the domain via the multiblock approach [23, 24, 33, 44, 48]. It offers sophisticated simulation components that encapsulate complex mathematical models of the physical interaction in the subsurface such as geomechanics and chemical processes, and which execute on parallel and distributed systems. Solvers employ state-of-the-art techniques for nonlinear and linear problems including Newton–Krylov methods enhanced with multigrid, two-stage, and physics-based preconditioners [20, 21]. It can also handle an arbitrary number of wells each with one or more completion intervals.

2.2. Economic model

In general, the economic value of production is a function of the time of production and of injection and production rates in the reservoir. It takes into account costs such as well drilling, oil prices, costs of injection, extraction, and disposal of water and of the hydrocarbons, as well as associated operating costs. We assume here that operation and drilling costs are independent of the well locations and therefore a constant part of our objective function that can be omitted for the purpose of optimization.

We then define our objective function by summing up, over the time horizon $[0, T]$, the revenues from produced oil over all production wells, and subtracting the costs of disposing produced water and the cost of injecting water. The result is the net present value (NPV) function

$$f_T(p) = -\int_0^T \left\{ \sum_{prod.\ wells} \left[\left(c_o q_o^-(t) - c_{w,disp} q_w^-(t) \right) \right] - \sum_{inj.\ wells} c_{w,inj} q_w^+(t) \right\} (1+r)^{-t}\, dt, \tag{3}$$

where q_o^- and q_w^- are production rates for oil and water, respectively, and q_w^+ are water injection rates, each in barrel/day. The coefficients c_o, $c_{w,disp}$ and $c_{w,inj}$ are the prices of oil and the costs of disposing and injecting water, in units of dollars per barrel each. The term r represents the interest rate per day, and the exponential factor takes into account that the drilling costs have to be paid up front and have to be paid off with interest. The function $f_T(p)$ is the negative total NPV, as our algorithms will be searching for a minimum; this then amounts to maximizing the (positive) revenue.

If no confusion is possible, we drop the subscript from f_T when we compare function evaluations for the same time horizon T but different well locations p. Note that $f(p)$ depends on the locations p of the wells in two ways. First, the injection rates of wells, and thus their associated costs, depend on their location if the BHP is prescribed. Second, the production rates of the wells as well as their water/oil ratio depend on where water is injected and where producers are located.

We remark that in practice, realistic objective functions would also include other factors that would render it more complex (see, e.g., [6, 9]). However, the general form would be the same.

2.3. The optimization problem

With the model and objective function described above, the optimization problem is stated as follows: *find optimal well locations* p^* *such that*

$$p^* = \arg\min_{p \in P} f(p), \tag{4}$$

subject to the flow variables used in $f(p)$ satisfying model (1), (2). The parameter space for optimization P is the set of possible well locations. We fix the BHP operating conditions at all wells. However, in general the BHP and possibly other parameters could vary and become an element of P as well.

If the model we consider are the continuous flow equations (1), (2), then P is the continuous set of well locations $p = (p_x, p_y) \in \mathbb{R}^2$. However, in practice, all we can consider is a discretized version of these equations, in particular the discrete solution provided by the IPARS flow solver mentioned above. Within this solver, a well model is used to describe fluid flow close to well locations, and the way this model is implemented implies that only the cell a well is in is relevant, whereas the location within a cell is ignored. Consequently, the search space P reduces to the set of cells, which we parameterize by the integer lattice of cell midpoints.

The optimization problem is therefore converted from a continuous one to a discrete one, and the optimization algorithms described below have to take this into account. While this transformation makes the problem more complicated to solve in general, it should be noted that it should not impact the solution process too much if the distance between adjacent points in P is less than the typical length scale of variation of the objective function $f(p)$, since in that case derivatives of the continuous function $f(p)$ can be well approximated by finite differences on the integer lattice. As will become obvious from figures 3 and 4 below, this assumption is satisfied in at least parts of the search space.

In the rest of the section, we briefly describe all the optimization algorithms that we compare for their performance on the problem outlined above. The implementation of these algorithms uses a Grid-enabled software framework previously described in [3, 19, 31].

2.3.1. *An integer SPSA algorithm*

The first optimization algorithm we consider is an integer version of the SPSA method. SPSA was first introduced by Spall [40, 42] and uses the following idea: in any given iteration, choose a random direction in search space. By using two evaluation points in this and the opposite direction, determine if the function values increase or decrease in this direction, and get an estimate of the value of the derivative of the objective function in this direction. Then take a step in the descent direction with a step length that is the product of the approximate value of the derivative, and a factor that decreases with successive iterations.

Appealing properties of this algorithm are that it uses only two function evaluations per iteration, and that each update is in a descent direction. In particular, the first of these properties makes it attractive compared to a standard finite difference approximation of the gradient of the objective function, which takes at least $N+1$ function evaluations for a function of N variables. Numerous improvements of this algorithm are possible, including computing approximations to the Hessian, and averaging over several random directions (see [42]).

In its basic form as outlined above, SPSA can only operate on unbounded continuous sets, and is thus unsuited for optimization on our bounded integer lattice P. A modified SPSA algorithm for such problems was first proposed and analyzed by Gerencsér et al. [10, 11]. While their method involved fixed gain step lengths and did not incorporate bounds, both points are easily integrated. In order to describe our algorithm, let us define $\lceil \cdot \rceil$ to be the operator that rounds a real number to the next integer of larger magnitude. Furthermore, let Π be the operator that maps every point outside the bounds of our optimization domain onto the closest point in P and does not modify points inside these bounds. Then the integer SPSA algorithm that we use for the computations in this paper is stated as follows (for simplicity of notation, we assume that we optimize over the lattice of all integers within the bounds described by Π, rather than an arbitrarily spaced lattice).

Algorithm 2.1 (Integer SPSA).

1 *Set* $k = 1, \gamma = 0.101, \alpha = 0.602$.
2 *While* $k < K_{\max}$ *or convergence has not been reached do*
  2.1 *Choose a random search direction* Δ_k, *with* $(\Delta_k)_l \in \{-1, +1\}, 1 \le l \le N$.
  2.2 *Compute* $c_k = \lceil \frac{c}{k^\gamma} \rceil, a_k = \frac{a}{k^\alpha}$.

2.3 *Evaluate* $f^+ = f(\Pi(p_k + c_k\Delta_k))$ *and* $f^- = f(\Pi(p_k - c_k\Delta_k))$.
2.4 *Compute an approximation to the magnitude of the gradient by* $g_k = (f^+ - f^-)/|\Pi(p_k + c_k\Delta_k) - \Pi(p_k - c_k\Delta_k)|$.
2.5 *Set* $p_{k+1} = \Pi(p_k - \lceil a_k g_k \rceil \Delta_k)$.
2.6 *Set* $k = k + 1$.

end While

A few comments are in order. The exponents γ and α control the speed with which the monotonically decreasing gain sequences defined in step 2.2 converge to zero, and consequently control the step lengths the algorithm takes. The sequence $\{a_k\}$ is required to decay at a faster rate than $\{c_k\}$ to prevent instabilities due to the numerical cancellation of significant digits. Therefore, $\alpha > \gamma$ to ensure $\sum_{k=0}^{\infty} a_k/c_k < \infty$. The values for γ and α listed in step 1 are taken from [41, 42], where a more in-depth discussion of these values can be found. The parameters c and a are chosen to scale step lengths against the size of the search space P.

As can be seen in step 2.1, the entries of the vector Δ_k are generated following a Bernoulli (± 1) distribution. This vector represents the simultaneous perturbation applied to all search space components in approximating the gradient at step 2.4. It can be shown that the expected value of this approximate gradient converges indeed to the true gradient for continuous problems.

Compared to the standard SPSA algorithm [42], our algorithm differs in the following steps: First, rounding the step length c_k in step 2.2 ensures that the function evaluations in step 2.3 only happen on lattice points (note that Δ_k is already a lattice vector). Likewise, the rounding operation in the update step 2.5 ensures that the new iterate is a lattice point. In both cases, we round up to the next integer to avoid zero step lengths that would stall the algorithm. Second, using the projector in steps 2.3–2.5 ensures that all iterates are within the bounds of our parameter space.

2.3.2. *An integer finite difference gradient algorithm*

The second algorithm is a finite difference gradient (FDG) algorithm that shares most of the properties of the SPSA algorithm discussed above. In particular, we use the same methods to determine the step lengths for function evaluation and iterate update (including the same constants for γ, α) and the same convergence criterion. However, instead of a random direction, we compute the search direction by a two-sided finite difference approximation of the gradient in a component-wise fashion.

Algorithm 2.2 (Integer Finite Difference Algorithm).

1 *Set* $k = 1, \gamma = 0.101, \alpha = 0.602$.
2 *While* $k < K_{\max}$ *or convergence has not been reached do*

2.1 *Compute* $c_k = \lceil \frac{c}{k^\gamma} \rceil, a_k = \frac{a}{k^\alpha}$.
2.2 *Set evaluation points* $s_i^{\pm} = \Pi(p_k \pm c_k e_i)$ *with* e_i *the unit vector in direction* $i = 1, \ldots, N$.
2.3 *Evaluate* $f_i^{\pm} = f(s_i^{\pm})$.
2.4 *Compute the gradient approximation* g *by* $g_i = (f_i^+ - f_i^-)/|s_i^+ - s_i^-|$.
2.5 *Set* $p_{k+1} = \Pi(p_k - \lceil a_k g \rceil)$.
2.6 *Set* $k = k + 1$.

end while

In step 2.5, the rounding operation $\lceil a_k g \rceil$ is understood to act on each component of the vector $a_k g$ separately. This algorithm requires $2N$ function evaluations per iteration, in contrast to the two evaluations in the SPSA method, but we can expect better search directions from it.

Note that this algorithm is closely related to the finite difference stochastic approximation (FDSA) algorithm proposed in [42]. In fact, in the absence of noise in the objective function, the two algorithms are the same.

2.3.3. *Very fast simulated annealing (VFSA)*

The third algorithm is the very fast simulated annealing (VFSA). This algorithm shares the property of other stochastic approximation algorithms in relying only on function evaluations. Simulated annealing attempts to mathematically capture the cooling process of a material by allowing random changes to the optimization parameters if this reduces the energy (objective function) of the system. Although the temperature is high, changes that increase the energy are also likely to be accepted, but as the system cools (anneals), such changes are less and less likely to be accepted.

Standard simulated annealing randomly samples the entire search space and moves to a new point if either the function value (i.e., the temperature or energy) is lower there; or, if it is higher, the new point is accepted with a certain probability that decreases over time (controlled by the temperature decreasing with time) and by the amount by which the new function value would be worse than the old one. On the other hand, VFSA also restricts the search space over time, by increasing the probability for sampling points closer rather than

farther away from the present point as the temperature decreases. The first of these two parts of VFSA ensures that as iterations proceed we are more likely to accept only steps that reduce the objective function, whereas the second part effectively limits the search to the local neighborhood of our present iterate as we approach convergence. The rates by which these two probabilities change are controlled by the "schedule" for the temperature parameter; this schedule is used for tuning the algorithm.

VFSA has been successfully used in several geophysical inversion applications [8, 39]. Alternative description of the algorithm can be found in [18].

2.3.4. *Other optimization methods*

In addition to the previous optimization algorithms, we have included two popular algorithms in some of the comparisons below: the Nelder–Mead (N–M) simplex algorithm and genetic algorithms (GA). Both of these approaches are gradient-free and are important algorithms for nonconvex or nondifferentiable objective functions. We will only provide a brief overview over their construction, and refer to the references listed below.

The Nelder–Mead algorithm (also called simplex or polytope algorithm) keeps its present state in the form of $N+1$ points for an N-dimensional search space. In each step, it tries to replace one of the points by a new one, by using the values of the objective function at the existing vertices to define a direction of likely decent. It then evaluates points along this direction and if it finds such a point subject to certain conditions, will use it to replace one of the vertices of the simplex. If no such point can be found, the entire simplex is shrunk. This procedure was employed in [5, 14] as part of a hybrid optimization strategy for the solution of the well placement problem. More information on the N–M algorithm can be found in [22] and in the original paper [28].

The general family of evolutionary algorithms is based on the idea of modeling the selection process of natural evolution. Starting from a number of "individuals" (points in search space), the next generation of individuals is obtained by eliminating the least fit (as measured by the objective function) and allowing the fittest to reproduce. Reproduction involves generating the "genome" of a child (a representation of its point in search space) by randomly picking elements of both parents' genome. In addition, random mutations are introduced to sample a larger part of search space.

The literature on GA is vast; here we only mention [12, 27]. In the context of the well placement problem, it has been one of the most popular optimization algorithms of choice; (see, e.g., [14, 47]) and the references therein.

In addition to the algorithms used in this study, there is a great number of other methods that may be applicable to the problem at hand, most notably Newton-type methods [29] or hybrid methods that are able to switch between stochastic and deterministic algorithms [8]. For brevity, we only consider the algorithms listed above, but will discuss the possible suitability of other algorithms in the Conclusions below.

3. Results for placement of a single well

3.1. The reservoir model

We consider a relatively simple 2D reservoir $\Omega = [0, 4880] \times [0, 5120]$ of roughly 25 million ft^2, which is discretized by 61×64 spatial grid blocks of 80 ft length along each horizontal direction, and a depth of 30 ft. Within this reservoir, we fix the location of five wells (two injectors and three producers; see figure 1) and want to optimize the location of a third injector well. Since the model consists of 3904 grid blocks, the set of possible well locations over which we optimize is the integer lattice $P = \{40, 120, 200, \ldots, 4840\} \times \{40, 120, 200, \ldots, 5080\}$ of cell midpoints. The reservoir under study is located at a depth of 1 km (3869 ft) and corresponds to a 2D section extracted from the Gulf of Mexico. The porosity has been fixed at $\phi = 0.2$ but the reservoir has a heterogeneous permeability field as shown in figure 1 The relative permeability and capillary pressure curves correspond to a single type

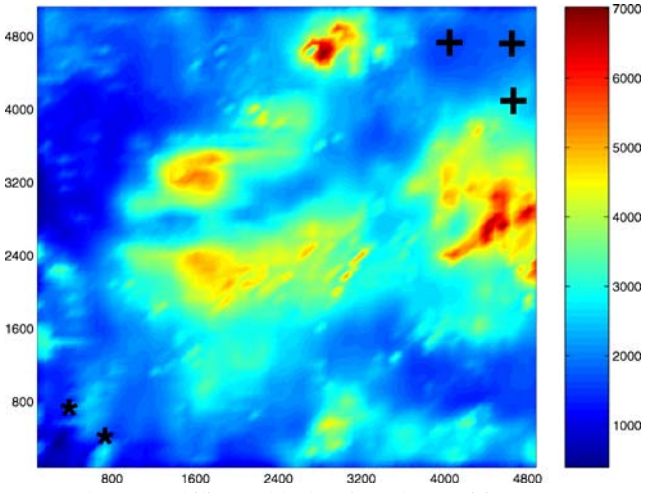

Figure 1 Permeability field showing the positions of current wells. The symbols "$*$" and "$+$" indicate injection and producer wells, respectively.

of rock. The reservoir is assumed to be surrounded by impermeable rock; that is, fluxes are zero at the boundary. The fluids are initially in equilibrium with water pressures set to 2600 psi and oil saturation to 0.7. For this initial case, an opposite-corner well distribution was defined as shown in figure 1.

For the objective function (3), cost coefficients were chosen as follows: $c_o = 24$, $c_{w,disp} = 1.5$ and $c_{w,inj} = 2$. We chose an interest rate r of $10\% = 0.1$ per year.

We undertook to generate a realistic data set for evaluation of optimization algorithms by computing economic revenues for all 3904 (i.e., $64\times$ 61) possible well locations, and for 200 different time horizons T between 200 and 2000 days. (The data for different time horizons can be obtained from the simulation by simply restricting the upper bound of the integral in (3).) Figure 2 shows the saturation and pressure field distribution after 2000 days of simulation. Plots of $f_T(p)$ for given values of T are shown in figures 3 and 4. As can be expected, the maxima move away from the producer wells as the time horizon grows, since close by injection wells may flood producer wells if the time horizon is chosen large enough, thus diminishing the economic return.

Each simulation for a particular p took approximately 20 min on a Linux PC with a 2-GHz AMD Athlon processor, for a total of 2000 CPU hours. Computations were performed in parallel on the Lonestar cluster at the Texas Advanced Computing Center (TACC). It is clear that, for more realistic computations, optimization algorithms have to rely only on single function evaluations, rather than evaluating the complete solution space. Hence, the number of function evaluations will be an important criterion in comparing the different optimization algorithms below.

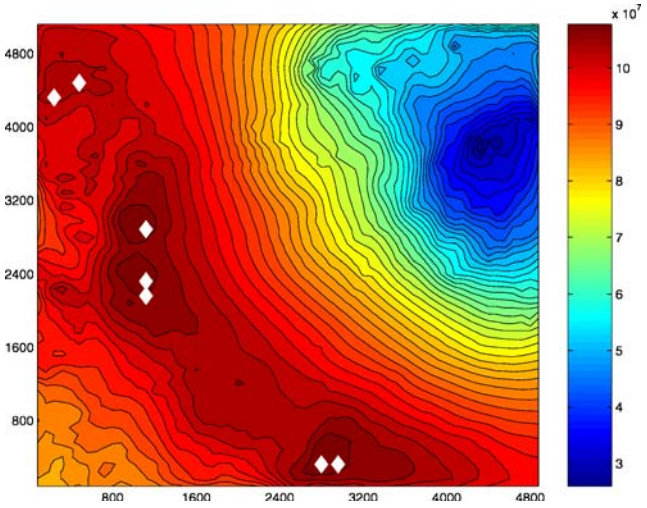

Figure 3 Search space response surface: expected (positive) revenue $-f_{2000}(p)$ for all possible well locations $p \in P$, as well as a few nearly optimal points found by SPSA.

Note that while it would in general be desirable to compute the global optimum, we will usually be content if the algorithm finds a solution that is almost as good. This is important in the present context, where the revenue surface plotted in figure 3 has 72 local optima, with the global optimum being $f(p = \{2920, 920\}) = -1.09804 \cdot 10^8$. However, there are five more local extrema within only half a percent of this optimal value, which makes finding the global optimum rather complicated.

Another reason to be content with good local optima is that, in general, our knowledge of a reservoir is incomplete. The actual function values and locations of optima are therefore subject to uncertainty, and it is more meaningful to ask for statistical properties of solutions found by optimization algorithms. Since the

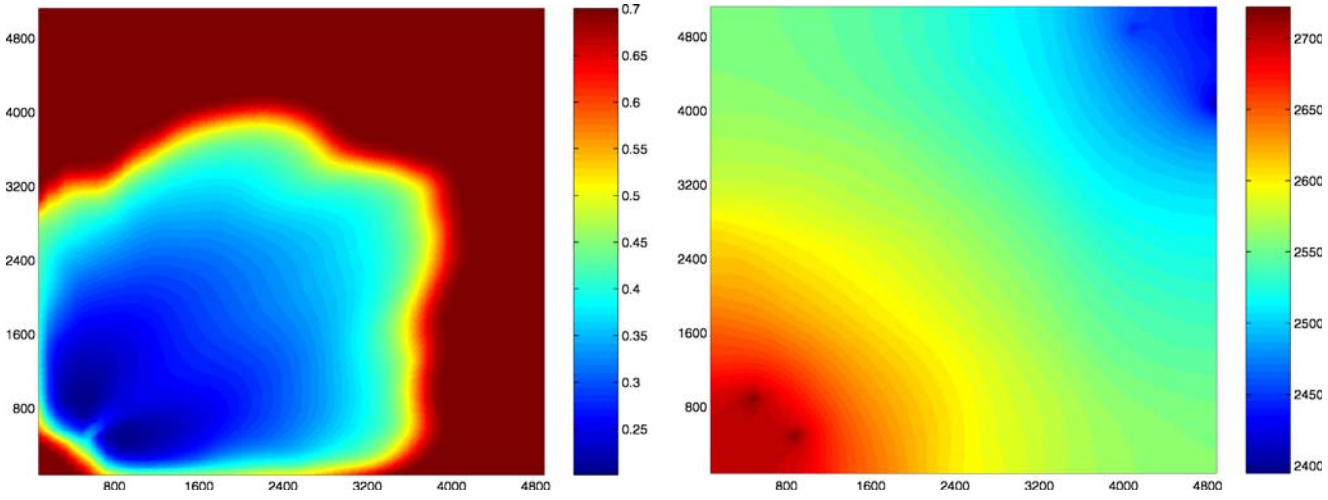

Figure 2 Left: Oil saturation at the end of the simulation for the original distribution of five wells. Right: Oil pressure.

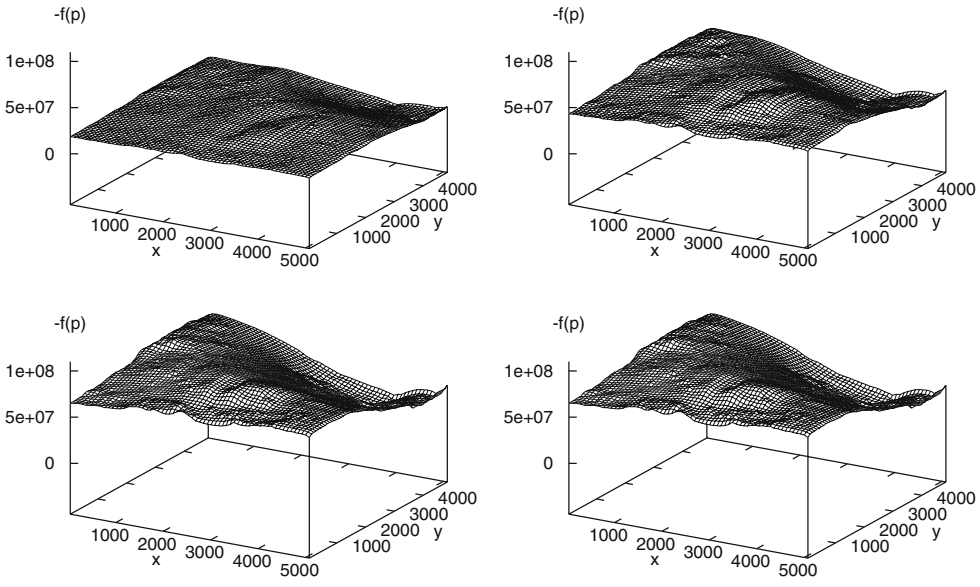

Figure 4 Surface plots of the (positive) revenue $-f_T(p)$ for $T = 500$ (top left), 1000 (top right), 1500 (bottom left), and 2000 (bottom right) days.

particular data set created for this paper is complete, we will therefore mostly be concerned with considering the results of running large numbers of optimization runs to infer how a single run with only a limited number of function evaluations would perform. In particular, we will compare the average quality of optima found by optimization algorithms with the global optimum, as well as the number of function evaluations the algorithms take.

3.2. Performance indicators

To compare the convergence properties of each of the optimization algorithms, three main performance indicators were considered: (1) effectiveness (how close the algorithm gets to the global minimum on average); (2) efficiency (running time) of an algorithm measured by the number of function evaluations required; (3) reliability of the algorithms, measured by the number of successes in finding the global minimum, or at least approaching it sufficiently close.

To compute these indicators, we started each algorithm from every possible location p_i in the set P, and for each of these N optimization runs record the point $\hat{p}_i$ where it terminated, the function value $f(\hat{p}_i)$ at this point, and the number K_i of function evaluations until the algorithm terminated. Since the algorithms may require multiple function evaluations at the same point, we also record the number L_i of *unique* function evaluations (function evaluations can be cached and repeated function evaluations are therefore inexpensive steps). Note that in this setting, we would have $\hat{p}_i = p^*$ for an algorithm that always finds the global optimum.

The effectiveness is then measured in terms of the average value of the best function evaluation

$$\bar{f} = \sum_{i=1}^{N} \frac{f(\hat{p}_i)}{N}. \tag{5}$$

and how close is this value to $f(p^*)$. The efficiency is given by the following two measures

$$\bar{K} = \sum_{i=1}^{N} \frac{K_i}{N}, \qquad \bar{L} = \sum_{i=1}^{N} \frac{L_i}{N}. \tag{6}$$

Finally, reliability or robustness can be expressed in terms of percentile values. A χ percentile is defined as the value that is exceeded by a fraction χ of results $f(\hat{p}_i)$; in particular, φ^{50} is the value such that $f(\hat{p}_i) \geq \varphi^{50}$ in 50% of runs (and similar for the 95 percentile φ^{95}).

3.3. Results for the integer SPSA algorithm

In all our examples, we choose $c = 5$, $a = 2 \cdot 10^{-5}$, and terminate the iteration if there is no progress over a number of iterations, measured by the criterion $|p_k - p_{k-\kappa}| < \xi$, where we set $\kappa = 6, \xi = 2$. All these quantities are stated for the unit integer lattice, and are appropriately scaled to the actual lattice P in our problem. Note that the values for a stated above lead to initial step lengths on the order of 20 on the unit lattice since $a = 20/\bar{g}$, where $\bar{g} = 10^8/100$ is an order-of-magnitude estimate of the size of gradients, with 10^8 being typical function values and 100 being roughly the diameter of the unit lattice domain.

The white marks in figure 3 indicate the best well positions found by the SPSA algorithm when started from

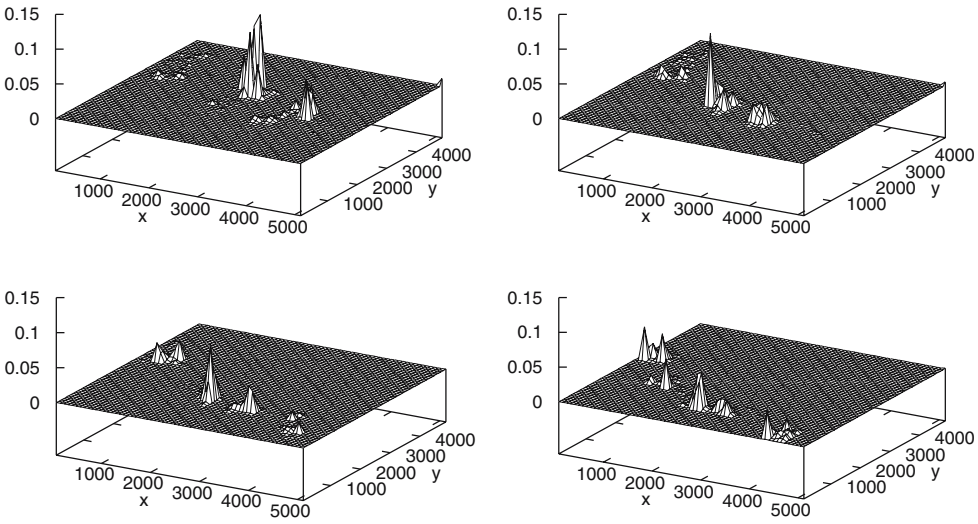

Figure 5 Probability surface for the SPSA algorithm stopping at a given point p for the corresponding objective functions shown in figure 4.

seven different points on the top-left to bottom-right diagonal of the domain. As can be seen, SPSA is able to find very good well locations from arbitrary starting points, even though there is no guarantee that it finds the global optimum every time. Note, however, that the SPSA algorithm can be modified to perform a global search by injected noise, although we did not pursue this idea here for the sake of algorithmic simplicity.

As stated above, the answers to both the effectiveness and reliability questions are closely related to the probability with which the algorithm terminates at any given point $p \in P$. For the three functions $f_{1000}(p)$, $f_{1500}(p)$, $f_{2000}(p)$ shown in figure 4, this probability of stopping at p is shown in figure 5. It is obvious that the locations where the algorithm stops are close to the (local) optima of the solution surface.

The statistical qualities of termination points of the integer SPSA algorithm are summarized in table 1 for the four data sets at $T = 500$, 1000, 1500, 2000 days. The table shows that on average the stopping position is only a few percent worse than the global optimum. The φ^{50} and φ^{95} values are important in the decision on how often to restart the optimization algorithm from different starting positions. Such restarts may be deemed necessary because the algorithm does not always stop at the global optimum, and we may want to improve on the result of a single run by starting from a different position and taking the better guess. Although the φ^{95} value reveals what value of $f(\hat{p}_i)$ we can expect from a single run in 95% of cases ("almost certainly"), the φ^{50} value indicates what value we can expect from the better of two runs started independently.

Despite the fact that both φ^{95} and φ^{50} are still relatively close to $f(p^*)$, the conclusion from these values is that to be within a few percent of the optimal value one run is not enough, while two are. Finally, the last two columns indicate that the algorithm, on average, only needs 37–52 function evaluations to converge; furthermore, if we cache previously computed values, only 30–42 function evaluations will be required. The worst of these numbers are for $T = 500$ days, where maxima are located in only a few places; the best results are for $T = 2000$ days, in which case maxima are distributed along a long arc across the reservoir domain.

3.4. Results for the integer FDG algorithm

Table 2 show the results obtained for the FDG algorithm described in section 2.3.2. From the table, it is obvious that for earlier times, the algorithm needs less function evaluations than SPSA. However, it also

Table 1 Statistical properties of termination points of the integer SPSA algorithm.

T	$f_T(p^*)$	$\bar{f}_T$	φ_T^{50}	φ_T^{95}	$\bar{K}_T$	$\bar{L}_T$
500	$-2.960 \cdot 10^7$	$-2.853 \cdot 10^7$	$-2.920 \cdot 10^7$	$-2.507 \cdot 10^7$	52.2	42.6
1000	$-6.696 \cdot 10^7$	$-6.412 \cdot 10^7$	$-6.490 \cdot 10^7$	$-5.834 \cdot 10^7$	41.0	33.1
1500	$-9.225 \cdot 10^7$	$-9.011 \cdot 10^7$	$-9.139 \cdot 10^7$	$-8.286 \cdot 10^7$	40.8	33.1
2000	$-1.098 \cdot 10^8$	$-1.075 \cdot 10^8$	$-1.086 \cdot 10^8$	$-1.046 \cdot 10^8$	37.8	30.2

Table 2 Statistical properties of termination points of the integer finite difference gradient algorithm.

T	$f_T(p^*)$	$\bar{f}_T$	φ_T^{50}	φ_T^{95}	$\bar{K}_T$	$\bar{L}_T$
500	$-2.960 \cdot 10^7$	$-2.708 \cdot 10^7$	$-2.794 \cdot 10^7$	$-2.232 \cdot 10^7$	52.6	24.4
1000	$-6.696 \cdot 10^7$	$-6.222 \cdot 10^7$	$-6.480 \cdot 10^7$	$-5.572 \cdot 10^7$	53.0	28.1
1500	$-9.225 \cdot 10^7$	$-8.837 \cdot 10^7$	$-9.133 \cdot 10^7$	$-8.211 \cdot 10^7$	55.5	32.4
2000	$-1.098 \cdot 10^8$	$-1.062 \cdot 10^8$	$-1.083 \cdot 10^8$	$-1.044 \cdot 10^8$	57.0	31.5

produces worse results, being significantly farther away from the global optimum on average. The reason for this is seen in figure 6, where we show the termination points of the algorithm: it is apparent that the algorithm quite frequently gets caught in local optima, at least much more frequently than the SPSA algorithm for which results are shown in figure 5. This leads to early termination of the algorithm and results in suboptimal overall results. Hence, as it is expected from a nonconvex or nondifferentiable objective function, accurate or noise-free gradient computations does not necessarily lead to a high level of effectiveness, efficiency, and reliability in the search for a (nearly) global optimal solution.

3.5. Results for VFSA

Table 3 and figure 7 show the results for the VFSA algorithm of section 2.3.3. As can be seen from the table, the results obtained with this algorithm are more reliable and effective than for the other methods, albeit at the cost of a higher number of function evaluations (i.e., poor efficiency). Nevertheless, these results can be further improved: by using a stretched cooling schedule (resulting in a further increase in the number of function evaluations), the algorithm found the global optimum in 95% of our runs even though only 10–15% of the elements of the search space are actually sampled. VFSA thus offers tuning possibilities for finding global optima that other, local algorithms do not have. The differences are particularly pronounced in the φ^{95} column, suggesting the more global character of the VFSA algorithms compared to the SPSA and FDG algorithms.

3.6. Other optimization algorithms and comparison of algorithms

For completeness, we include comparisons with the Nelder–Mead (N–M) and genetic algorithm (GA) approaches. Using the revenue surface information defined on each of the 3904 points, we were able to test these algorithms without relying on further IPARS runs. Since the efficiency is measured in terms of function evaluations, CPU time was not a major concern and additional experiments turn out to be affordable in other computational settings such as Matlab.

The Nelder–Mead (N–M) algorithm runs were based on the Matlab intrinsic function `fminsearch`. For the optional argument list, we set the nonlinear tolerance to 0.4 and defined the maximum number of iterations to 100. For the GA tests, we relied on the GAOT toolbox developed by Houck et al. [16]. The `ga` function of this toolbox allows the definition of different starting

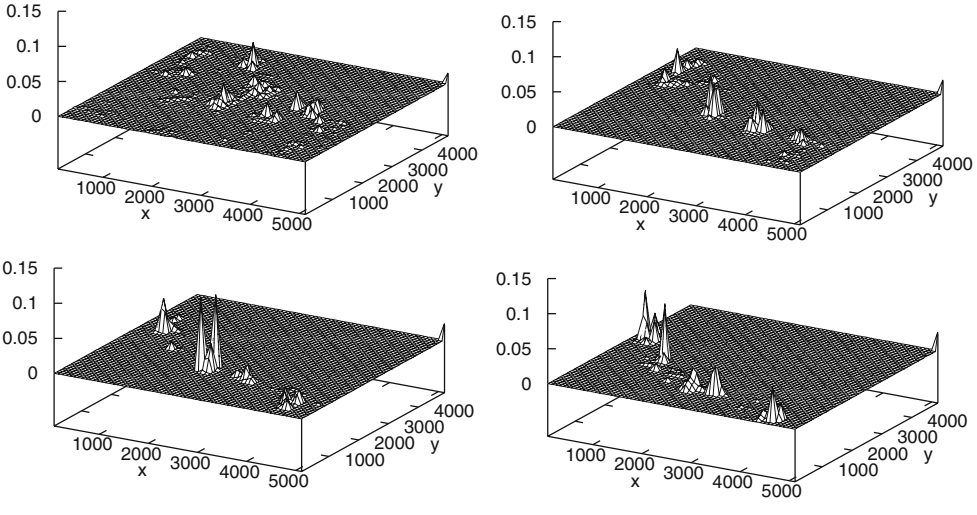

Figure 6 Probability surface for the FDG algorithm stopping at a given point p for the corresponding objective functions shown in figure 4.

Table 3 Statistical properties of termination points of the VFSA algorithm.

T	$f_T(p^*)$	$\bar{f}_T$	φ_T^{50}	φ_T^{95}	$\bar{K}_T$	$\bar{L}_T$
500	$-2.960 \cdot 10^7$	$-2.842 \cdot 10^7$	$-2.854 \cdot 10^7$	$-2.731 \cdot 10^7$	79.3	66.7
1000	$-6.696 \cdot 10^7$	$-6.556 \cdot 10^7$	$-6.601 \cdot 10^7$	$-6.423 \cdot 10^7$	68.9	60.0
1500	$-9.225 \cdot 10^7$	$-9.907 \cdot 10^7$	$-9.142 \cdot 10^7$	$-8.863 \cdot 10^7$	78.0	66.0
2000	$-1.098 \cdot 10^8$	$-1.083 \cdot 10^8$	$-1.083 \cdot 10^8$	$-1.051 \cdot 10^8$	75.5	63.9

populations, crossover, and mutation functions among other amenities. In our runs, we specified an initial population of eight individuals, a maximum number of 40 generations, and the same nonlinear tolerance as specified for the N–M algorithm. Since, we always started with eight individuals, the algorithm was rerun 488 times to reproduce the coverage of the 3904 different points. The rest of the `ga` arguments were set to the default values. In both cases, the implementations only provided the number of function evaluations K_i, but not the number of unique evaluations L_i; therefore, the comparison will only be made with the former.

Table 4 summarizes the comparison for all optimization algorithms for $T = 2000$ days. Clearly, the N–M turns out to be the worst one since it frequently hit the maximum number of function evaluations without retrieving competitive solutions to the other approaches. This is mainly due to the fact that the simplex tended to shrink very fast and the algorithm got prematurely stuck without approaching any local or global minimum. The GA shows a more effective and reliable solution but still falls short of the numbers displayed by both the SPSA and VFSA algorithms. It is the least efficient one and shows difficulties in finding an improved solution after a few trials, as the φ^{50} column indicates. Slight improvements were found by increasing the initial population and tuning the rate of mutations; however, the authors found that the tuning of this algorithm was not particularly obvious for this particular problem.

As conclusion of this section, we state that the SPSA algorithm is the most efficient one in finding good solutions. VFSA can be made to find even better solutions, but it requires significantly more function evaluations to do so.

4. Results for placing multiple wells

In the previous section, we have considered optimizing the placement of a single well. The case was simple enough that we could exhaustively evaluate all elements of the search space in order to find the global maximum and to evaluate how different optimization algorithms perform. In this section, we consider the more complicated case that we want to optimize the placement of several wells at once. Based on the previous results and on computing time limitations, we restrict our attention to the SPSA, FDG, and VFSA algorithms. The parameters defined for each of them remained unchanged.

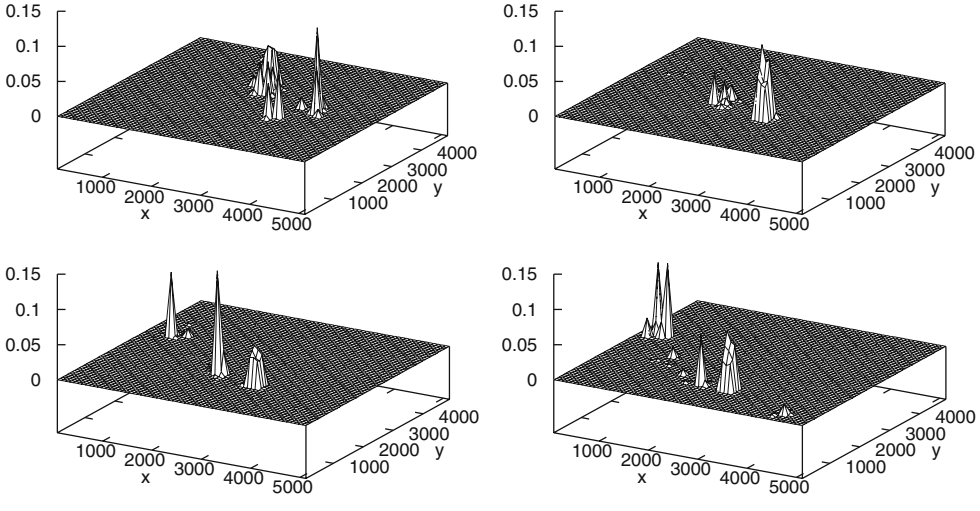

Figure 7 Probability surface for the VFSA algorithm stopping at a given point p for the corresponding four objective functions shown in figure 4.

Table 4 Comparison of different optimization algorithms for the objective function at $T = 2000$ days.

Method	$\bar{f}_T$	φ_T^{50}	φ_T^{95}	$\bar{K}_T$
$N-M$	$-1.018 \cdot 10^8$	$-1.078 \cdot 10^8$	$-8.977 \cdot 10^7$	99.9
GA	$-1.073 \cdot 10^8$	$-1.072 \cdot 10^8$	$-1.047 \cdot 10^8$	104.0
$VFSA$	$-1.083 \cdot 10^8$	$-1.083 \cdot 10^8$	$-1.051 \cdot 10^8$	75.5
FDG	$-1.062 \cdot 10^8$	$-1.083 \cdot 10^8$	$-1.044 \cdot 10^8$	57.0
$SPSA$	$-1.075 \cdot 10^8$	$-1.086 \cdot 10^8$	$-1.046 \cdot 10^8$	37.8

4.1. Placing two wells at once

In an initial test, we tried to place not only one but two new injector wells into the reservoir, for a total of four injectors and three producers. We compared our optimization algorithms for this, now four-dimensional, problem with the following simple strategy: take the locations of the five fixed wells used in the last section, and place a third injector at the optimal location determined $\{x_1, y_1\} = \{2920, 920\}$ in the previous section; then find the optimal location for a fourth injector using the same techniques. Thus, we solved a sequence of two two-dimensional problems instead of the four-dimensional one.

The search space response surface for placing the fourth well is shown in figure 8. In addition to the dips in all the same places as before, the surface has an additional depression in the vicinity of where the third injector was placed, indicating that the fourth injector should not be too close to the third one. The optimal location for this fourth injector is $\{x_2, y_2\} = \{200, 2680\}$, with an integrated revenue of $f(p) = -1.24175 \cdot 10^8$ at $p = \{2920, 920, 200, 2680\}$, i.e., roughly 13% better than the result with only three injectors.

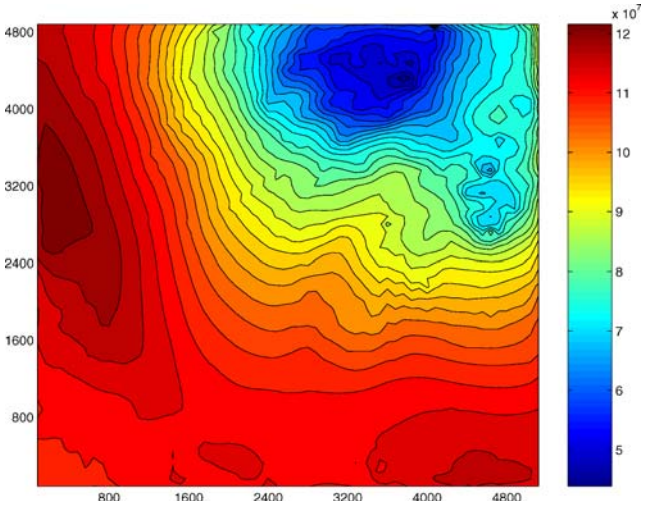

Figure 8 Search space response surface for placing a fourth injector when keeping the positions of the other six wells fixed.

It is obvious that the optimum of the full, four-dimensional problem must be at least as good as that of the simpler, sequential one. However, despite extensive computations and finding the above locations multiple times, we were unable to find positions for the two wells that would yield a better result than that given above. We believe that by mere chance, the simplified problem may have the same optimum as the full one. We therefore turn to a more challenging problem of optimizing the locations of all seven wells at once, which we describe in the next section. However, it is worth noting that a sequential well placement approach as above may be useful to obtain a good initial guess for the simultaneous well placement.

4.2. Placing seven wells at once

As a final example, we consider the simultaneous placement of all four injectors and three producers at once, i.e., without fixing the positions of any of them. The search space for this problem has, taking into account symmetries due to identical results when exchanging wells of the same type, approximately $8.55 \cdot 10^{22}$ elements, clearly far too many for an exhaustive search.

The results for this show a clear improvement over the well locations derived above: the best set of well locations found generates integrated revenues up to $f(p) = -1.65044 \cdot 10^8$, i.e., more than 30% better than the best result presented in the last section. We started 20 SPSA runs from different random arrangements of wells; out of these, 11 runs reached function values of $-1.5 \cdot 10^8$ or better within the first 150 function evaluations. The respective best well location arrangements together with their function values for the best four runs are shown in figure 9. All these well arrangements clearly make sense, with producers and injectors separated, and injectors aligned into patterns that form lines driving oil into the producers. In fact, the arrangements resemble standard patterns for producer and injector lineups used in practice (see, e.g., [17]). It should be noted that each of these arrangements likely corresponds to a local extremum of the objective function and that further iterations will not be able to morph one of the patterns into another without significantly pessimizing the objective function in-between.

Figure 10 documents the SPSA run that led to the discovery of the best well location shown at the left part of figure 9. To see how the algorithm proceeds for large numbers of iterations, we have switched off the convergence criterion that would otherwise have stopped the algorithm after approximately 80 iterations. From the

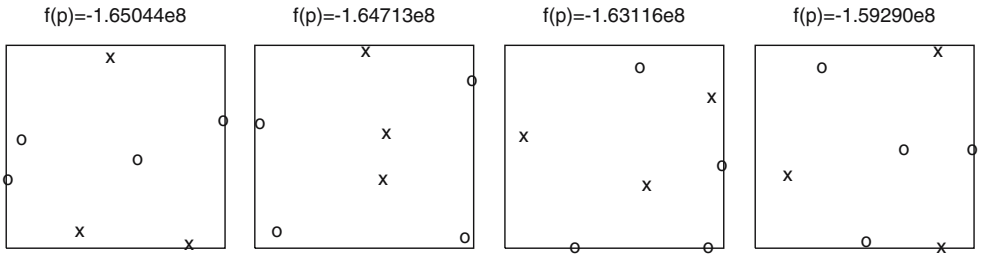

Figure 9 The best well arrangements found by the best four runs of SPSA from random locations, along with their revenue values. Producer well locations are marked with "×," injectors with "∘."

left part of the figure, it is apparent that the algorithm would have basically converged at that time (the best point found up to then, in function evaluation 72, is only half a percent worse than the best point found later in evaluation 191), and that good arrangements were already found after as little as 50 function evaluations. This has to be contrasted to the FDG algorithm that already takes at least 15 (for the one-sided derivative approximation) or even 28 (for the two-sided derivative approximation) function evaluations in each single iteration for this 14-dimensional problem.

The right part of figure 10 shows the locations of the seven wells at various iterations. Figure 11 also shows the corresponding water saturations at the end of the simulation period of 2000 days for four of these configurations. Obviously, the main reason that SPSA does not find any well configurations that are better than the one shown can be qualitatively visualized from the amount of waterflooding covering the reservoir, so that no additional revenue is possible: the well configuration found effectively drains the reservoir completely within the given time horizon.

4.2.1. *Comparison with the integer FDG algorithm*

Compared to the SPSA results above, the FDG algorithm performs significantly worse: only 7 out of 20 runs started at the same random locations reached a function value of $-1.5 \cdot 10^8$ or better within the first 150 function evaluations. In addition, the best function evaluation had a value of $-1.6406 \cdot 10^8$, roughly 0.6% worse than the best result obtained by SPSA. The resulting configuration is almost identical to the last one shown in the rightmost panel of figure 9. The values of the objective function encountered during this run are shown in the left panel of figure 12, to be compared with figure 10. It is obvious that the FDG algorithm requires significantly more function evaluations before it gets close to its convergence point. The steps in the graph are caused by the fact that the algorithm samples $2N = 28$ points

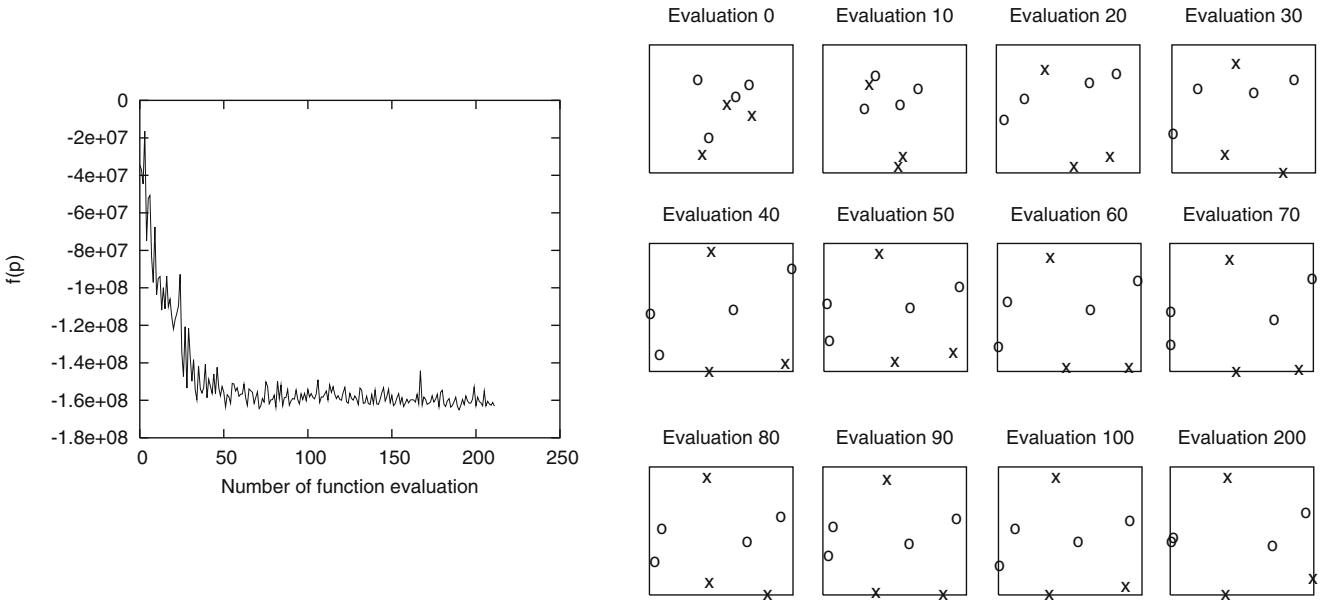

Figure 10 Progress of the SPSA run that led to the well location shown at the left of figure 9. Left: Reduction of objective function as iterations proceed. Right: Corresponding well arrangements at selected iterations.

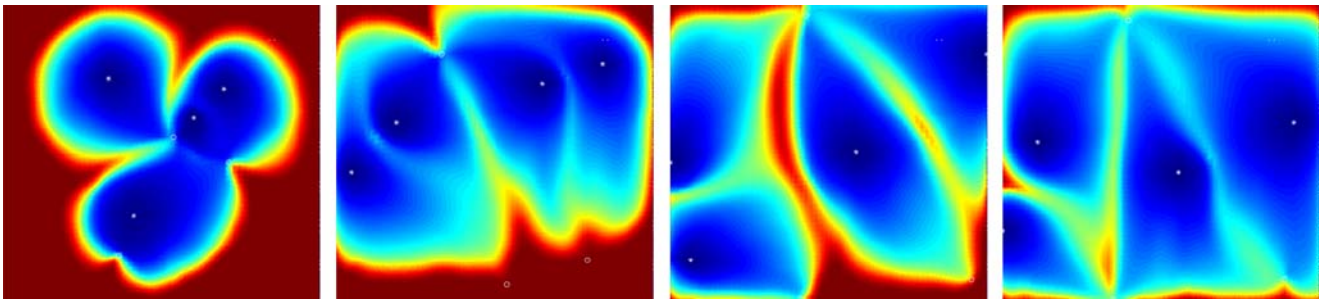

Figure 11 Oil saturations at the end of the simulation for the well configurations of figure 10 at function evaluations 0, 20, 40, and 100. Asterisks: injector wells; circles: producers. Saturations are between 0.2 (blue) and 0.7 (brown).

in each iteration before it moves on to the next iterate; these 28 points are all located close to each other and have therefore similar function values.

4.3. Comparison with the VFSA algorithm

The VFSA algorithm performed very well. In addition to 13 out of 20 runs that reached a best function value of $-1.5 \cdot 10^8$ or better within the first 150 function evaluations, it also scored the best function value found in all our runs with $f(p) = -1.65356 \cdot 10^8$. This best run is documented in the right panel of figure 12. The raggedness of the plot comes from the fact that VFSA does not only sample close-by points, but also ones that are farther away and therefore have significantly different function values than the presently best point. The well pattern corresponding to the best solution is again similar to the leftmost one shown in figure 9.

In comparison with the other algorithms discussed above, VSFA finds solutions that are at least as good, and finds them with a high frequency. However, it needs more function evaluations than SPSA to get to function values that are close to the very best found in our experiments.

5. Conclusions and Outlook

In this work, we investigated the performance of different optimization algorithms for the problem of placing wells in an oil reservoir. The testcases we have considered are (1) the two-dimensional problem of placing one well into a reservoir in which six others have already been placed at fixed positions, and (2) the 14-dimensional problem of determining the best positions of all seven of these wells at the same time. For the first of these problems, all possible well locations have been exhaustively explored in order to determine the statistical properties of optimization algorithms when started from randomly chosen locations. This allowed us to compare how algorithms perform on average, when no prior knowledge is available to determine good starting positions for the algorithms, as well as near-best and near-worst case behaviors. While the virtue of the second testcase obviously is its complexity, the strength of the first is that we know

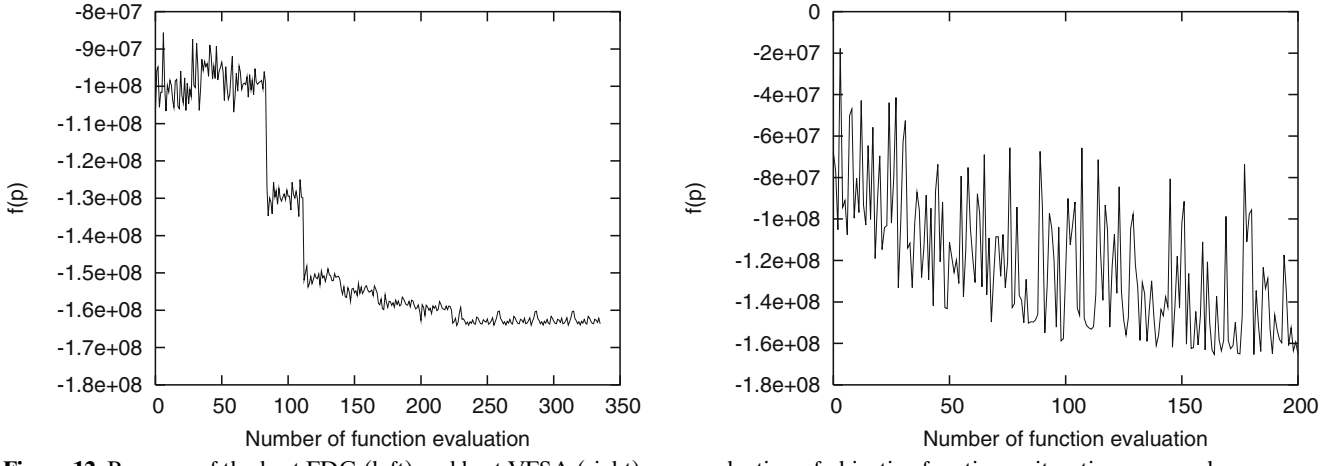

Figure 12 Progress of the best FDG (left) and best VFSA (right) runs: reduction of objective function as iterations proceed.

the exact optimum and all other function values, which allows us to do in-depth comparisons of the different algorithms.

On these testcases, we compared the simultaneous perturbation stochastic approximation (SPSA), finite difference gradient (FDG), very fast simulated annealing (VFSA), and, for the first testcase, the Nelder–Mead simplex and a genetic algorithm (GA) optimization methods. For the rather simple first testcase, both SPSA and VFSA reliably found very good solutions, and are significantly better than all the other algorithms; however, their different properties allow them to be tailored to different situations: while SPSA was very efficient in finding good solutions with a minimum of function evaluations, VFSA can be tweaked to find the global optimum almost always, at the expense of more function evaluations.

These observations also hold for the second, significantly more complex testcase. There, both SPSA and VFSA performed markedly better than the FDG algorithm. Both algorithms found good solutions in most of the runs that were started from different initial positions. Again, SPSA was more efficient in finding good positions for the seven wells in very few (around 50) function evaluations, while VFSA obtained good locations more reliably but with more function evaluations. FDG performed worse than the two other algorithms mainly because the high dimensionality of the problem requires too many function evaluations per iteration to let FDG be competitive.

In this paper, we have only considered a limited number of methods; in particular, we have omitted Newton-type or other algorithms using second-derivative information. These may be subject of further studies. However, we consider it uncertain whether they would be able provide superior performance: first, the objective functions considered here have many minima, maxima, and saddlepoints (in particular it is not convex), and second-order algorithms will only be able to perform well if curvature information is properly accounted for. Second, the objective surfaces are very rough in some parts of the parameter space (see figures 3 and 4), particularly in the vicinity of the extrema, with roughness length scales on the same order of magnitude as the spacing of the integer lattice on which we optimize; it is therefore already hard to extract gradient information, as witnessed by the consistently worse performance of the finite difference gradient method, and second derivative information will hardly contain much global information. Finally, in order to be superior, such algorithms need to be content with very few iterations. For example, the NEWUOA algorithm [36] for unconstrained optimization without derivatives requires a recommended number of 5 function evaluations per iteration for the single well problem, and 29 for the seven-well problem. On the other hand, our best algorithms only require on average 30–50 and 200–250 function evaluations, respectively, which would mean that the Newton-type algorithms would have to converge in 6–10 iterations; although possible, it seems unlikely that algorithms can achieve or improve on this given the roughness of the solution surface.

While the results presented in this work are a good indication as to which algorithms will perform well on more complex problems, more experiments are needed in this direction. In particular, the fact that we have only chosen the location of wells as decision variables is not sufficient to model the reality of oil reservoir management. For this, we would also have to allow for varying pumping rates, different and more complex well types, and different completion times of wells. In addition, constraints need to be placed on the system, such as maximal and minimal capacities of surface facilities, and the reservoir description must be more complex than the relatively simple 2D case we chose here in order to keep computing time within a range where different algorithms can be easily compared. Also, it will be interesting to investigate the effects of uncertainty in the reservoir description on the performance of algorithms; for example, we expect that averaging over several possible realizations of a stochastic reservoir model may smooth out the objective function, which would probably aid the FDG algorithm more than the other methods. Research in these directions is presently underway, and we will continue to investigate which algorithms are best suited for more complex and more realistic descriptions of the oil well placement problem.

Acknowledgements The authors want to thank the National Science Foundation (NSF) for its support under the ITR grant EIA 0121523/EIA-0120934.

References

1. Arbogast, T., Wheeler, M.F., Yotov, I.: Mixed finite elements for elliptic problems with tensor coefficients as cell-centered finite differences. SIAM, J. Numer. Anal. **34**(2), 828–852 (1997)
2. Aziz, K., Settari, A.: Petroleum reservoir simulation. Applied Science, London (1979)
3. Bangerth, W., Klie, H., Matossian, V., Parashar, M., Wheeler. M.F.: An autonomic reservoir framework for the stochastic optimization of well placement. Cluster Computing **8**(4), 255–269 (2005)
4. Becker, B.L., Song, X.: Field development planning using simulated annealing-optimal economic well scheduling and

placement. In: SPE Annual Technical Conference and Exhibition, Dallas, Texas, SPE 30650, October 1995
5. Bittencourt, A.C., Horne, R.N.: Reservoir development and design optimization. In SPE Annual Technical Conference and Exhibition, San Antonio, Texas, SPE 38895, October 1997
6. Carlson, M.: Practical reservoir simulation. PennWell Corporation (2003)
7. Centilmen, A., Ertekin, T., Grader, A.S.: Applications of neural networks in multiwell field development. In: SPE Annual Technical Conference and Exhibition, Dallas, Texas, SPE 56433, October 1999
8. Chunduru, R.K., Sen, M., Stoffa, P.: Hybrid optimization methods for geophysical inversion. Geophysics **62**, 1196–1207 (1997)
9. Fanchi, J.R.: Principles of Applied Reservoir Simulation. Boston: Butterworth-Heinemann Gulf Professional Publishing, 2nd edn. (2001)
10. Gerencsér, L., Hill, S.D., Vágó, Z.: Optimization over discrete sets via SPSA. In Proceedings of the 38th Conference on Decision and Control, Phoenix, AZ, 1999, 1791–1795 (1999)
11. Gerencsér, L., Hill, S.D., Vágó, Z.: Discrete optimization via SPSA. In: Proceedings of the Americal Control Conference, Arlington, VA, 2001, 1503–1504 (2001)
12. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)
13. Guyaguler, B.: Optimization of well placement and assessment of uncertainty. PhD thesis, Stanford University, Department of Petroleum Engineering (2002)
14. Guyaguler, B., Horne, R.N.: Uncertainty assessment of well placement optimization. In: SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, September, October 2001. SPE 71625
15. Helmig, R.: Multiphase Flow and Transport Processes in the Subsurface. Springer, Berlin (1997)
16. Houck, C., Joines, J.A., Kay, M.G.: A genetic algorithm for function optimization: A Matlab implementation. Technical Report TR 95-09, North Carolina State University (1995)
17. Hyne, N.: Nontechnical Guide to Petroleum Geology, Exploration, Drilling and Production. Pennwell Books, 2nd edn. (2001)
18. Ingber, L.: Very fast simulated reannealing. Math. Comput. Model. **12**, 967–993 (1989)
19. Klie, H., Bangerth, W., Wheeler, M.F., Parashar, M., Matossian, V.: Parallel well location optimization using stochastic algorithms on the grid computational framework. In: 9th European Conference on the Mathematics of Oil Recovery, ECMOR, Cannes, France, EAGE August 30–September 2, 2004
20. Lacroix, S., Vassilevski, Y., Wheeler, J., Wheeler, M.F.: Iterative solution methods for modelling multiphase flow in porous media fully implicitly. SIAM J. Sci. Comput. **25**(3), 905–926 (2003)
21. Lacroix, S., Vassilevski, Y., Wheeler, M.F.: Iterative solvers of the Implicit Parallel Accurate Reservoir Simulator (IPARS). Numer. Linear Algebra Appl. **4**, 537–549 (2001)
22. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence behavior of the Nelder–Mead simplex algorithm in low dimensions. SIAM J. Optim. **9**, 112–147 (1999)
23. Lu, Q.: A parallel multi-block/multi-physics approach for multi-phase flow in porous media. PhD thesis, University of Texas at Austin, Austin, Texas (2000)
24. Lu, Q., Peszyńska, M., Wheeler, M.F.: A parallel multi-block black-oil model in multi-model implementation. In: 2001 SPE Reservoir Simulation Symposium, Houston, Texas, SPE 66359 (2001)
25. Lu, Q., Peszyńska, M., Wheeler, M.F.: A parallel multi-block black-oil model in multi-model implementation. SPE J. **7**(3), 278–287 SPE 79535 (2002)
26. Mattax, C.C., Dalton, R.L.: Reservoir simulation. In: SPE Monograph Series, volume 13, Richardson, Texas (1990)
27. Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press (1996)
28. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
29. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research. Springer, New York (1999)
30. Pan, Y., Horne, R.N.: Improved methods for multivariate optimization of field development scheduling and well placement design. In: SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, 27–30, SPE 49055 September 1998
31. Parashar, M., Klie, H., Catalyurek, U., Kurc, T., Bangerth, W., Matossian, V., Saltz, J., Wheeler, M.F.: Application of grid-enabled technologies for solving optimization problems in data-driven reservoir studies. Future Gener. Comput. Syst. **21**(1), 19–26 (2005)
32. Parashar, M., Wheeler, J.A., Pope, G., Wang, K., Wang, P.: A new generation EOS compositional reservoir simulator. Part II: Framework and multiprocessing. In: Fourteenth SPE Symposium on Reservoir Simulation, Dallas, Texas, 31–38. Society of Petroleum Engineers June 1997
33. Parashar, M., Yotov, I.: An environment for parallel multiblock, multi-resolution reservoir simulations. In: Proceedings of the 11th International Conference on Parallel and Distributed Computing and Systems (PDCS 98), 230–235, Chicago, IL, International Society for Computers and their Applications (ISCA) Sep. 1998
34. Pardalos, P.M., Resende, M.G.C. eds.: Handbook of Applied Optimization, pp. 808–813. Oxford University Press (2002)
35. Peszyńska, M., Lu, Q., Wheeler, M.F.: Multiphysics coupling of codes. In: L.R. Bentley, J.F. Sykes, C.A. Brebbia, W.G. Gray, G.F. Pinder, eds., Computational Methods in Water Resources, 175–182. A. A. Balkema (2000)
36. Powell, M.J.D.: The NEWUOA software for unconstrained optimization without derivatives. Technical Report DAMTP 2004/NA08, University of Cambridge, England (2004)
37. Rian, D.T., Hage, A.: Automatic optimization of well locations in a north sea fractured chalk reservoir using a front tracking reservoir simulator. In: SPE International Petroleum & Exhibition of Mexico, Veracruz, Mexico, SPE 28716, October 2001
38. Russell, T.F., Wheeler, M.F.: Finite element and finite difference methods for continuous flows in porous media. In: R.E. Ewing, eds., The Mathematics of Reservoir Simulation, pp. 35–106. SIAM, Philadelphia (1983)
39. Sen, M., Stoffa, P.: Global Optimization Methods in Geophysical Inversion. Elsevier (1995)
40. Spall, J.C.: Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Trans. Autom. Control. **37**, 332–341 (1992)
41. Spall, J.C.: Adaptive stochastic approximation by the simultaneous perturbation method. IEEE Trans. Autom. Contr. **45**, 1839–853 (2000)
42. J.C. Spall. Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control. Wiley, New Jersey (2003)
43. Wang, P., Yotov, I., Wheeler, M.F., Arbogast, T., Dawson, C.N., Parashar, M., Sepehrnoori, K.: A new generation EOS compositional reservoir simulator. Part I: Formulation and discretization. In: Fourteenth SPE Symposium on Reservoir

Simulation, Dallas, Texas, pp. 55–64. Society of Petroleum Engineers, June 1997
44. Wheeler, M.F.: Advanced techniques and algorithms for reservoir simulation, II: The multiblock approach in the integrated parallel accurate reservoir simulator (IPARS). In: J. Chadam, A. Cunningham, R.E. Ewing, P. Ortoleva, M.F. Wheeler, eds., IMA Volumes in Mathematics and its Applications, Volume 131: Resource Recovery, Confinement, and Remediation of Environmental Hazards. Springer (2002)
45. Wheeler, M.F., Wheeler, J.A., Peszyńska, M.: A distributed computing portal for coupling multi-physics and multiple domains in porous media. In: L.R. Bentley, J.F. Sykes, C.A. Brebbia, W.G. Gray, G.F. Pinder, eds., Computational Methods in Water Resources, 167–174. A. A. Balkema (2000)
46. Yeten, B.: Optimum deployment of nonconventional wells. PhD thesis, Stanford University, Department of Petroleum Engineering (2003)
47. Yeten, B., Durlofsky, L.J., Aziz, K.: Optimization of nonconventional well type, location, and trajectory. SPE J. **8**(3), 200–210 SPE 86880 (2003)
48. Yotov, I.: Mortar mixed finite element methods on irregular multiblock domains. In: J. Wang, M.B. Allen, B. Chen, T. Mathew, eds., Iterative Methods in Scientific Computation, IMACS series Comp. Appl. Math., vol. 4, 239–244. IMACS, (1998)
49. Zhang, F., Reynolds, A.: Optimization algorithms for automatic history matching of production data. In: 8th European Conference on the Mathematics of Oil Recovery, ECMOR, Freiberg, Germany, EAGE, September 2002