PARALLEL WELL LOCATION OPTIMIZATION USING STOCHASTIC ALGORITHMS ON THE GRID COMPUTATIONAL FRAMEWORK

Hector Klie[1], Wolfgang Bangerth[1,2], Mary F. Wheeler[1], Manish Parashar[3] and Vincent Matossian[3]

[1]*Center for Subsurface Modeling, The University of Texas at Austin, Austin, TX.*
[2]*Institute for Geophysics, The University of Texas at Austin, Austin, TX.*
[3]*The Applied Software Systems Laboratory, Rutgers University, Piscataway, NJ.*

Abstract
The determination of optimal well locations is a challenging problem in oil production since it depends on geological and fluid properties as well as on economic parameters. This work addresses the efficient solution of this problem by using advanced techniques for coupling three important components of autonomic optimization: the Integrated Parallel Accurate Reservoir Simulator (IPARS) for production prediction, new optimization algorithms, in particular the Simultaneous Perturbation Stochastic Approximation (SPSA) approach, and the Grid infrastructure to access computational resources on the network in a seamless way. We illustrate the methodology using numerical results based on real data.

1. Introduction

Optimizing how and where wells are drilled in an oil reservoir is a problem with both high economic impact and high complexity. Traditionally, this task is carried out by analyzing a few scenarios with a numerical reservoir simulator. However, this may potentially result into misleading decisions with large consequences. Optimization algorithms promise to perform a systematic exploration of a broader set of scenarios and aim at finding the optimum under some given conditions. Together with the experience of specialists, they also allow for a better assessment of uncertainty and reduction of risk in decision-making. The main constraint for their use is the cost of repeatedly evaluating different exploitation scenarios via numerical solution of a complex set of coupled nonlinear partial differential equations on up to millions of gridblocks.

In this study, we present a computational framework for this problem, including Grid enabled technologies [4] to automatically discover and use available computational resources in a distributed environment, efficient optimization algorithms, and an efficient reservoir simulator. Using this framework, we have generated a complete dataset for a small but realistic test case in which only the two horizontal co-ordinates of a vertical well are subject to optimization. This large dataset is at the outer reaches of what is presently feasible with today's computational facilities. The test case shows a number of pathologies that also need to be expected in more realistic problems, including a large number of local optima. In this paper, we use it to test and explore how two different optimization algorithms perform.

The optimization algorithms we present in this paper are a version of the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm [8] that was modified to handle the fact that the optimization domain is a bounded integer lattice, and a version of a Finite Difference Gradient algorithm adapted to the same constraints. As will be shown, both algorithms need about as many function evaluations, but SPSA is significantly less prone to get caught in local maxima, and is thus a more reliable tool for optimization of problems with many local extrema.

The software used for these computations builds on the DISCOVER toolkit which allows agents running on geographically distributed machines to communicate with each other, to detect avail-

able resources, and to run reservoir simulations on them [2]. It is thus an ideal tool for applications that potentially run for very long times and that need to make use of computational resources with availability and capabilities that may change over time.
The paper is organized as follows. In the following section, we present the well placement problem in terms of the reservoir model and the economic model we use as an objective function. We also introduce the two optimization algorithms we will use on this problem. In Section 3, we give an overview of the Grid technology on which our software is built. Numerical results are then presented in Section 4, before concluding with Section 5.

2. Description of the problem

In this section, we provide a brief overview of the oil-water reservoir model within the IPARS framework. We then proceed to describe the objective function for the optimization problem. The last part of the section is devoted to integer versions of the SPSA and FDG optimization algorithms.

2.1 The reservoir flow model

For the purpose of this work, we describe the reservoir model by a set of partial differential equations for the conservation of mass of each component $m=o,w$ (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 the concentration of a phase m, and q_m the sources (production and injection rates). The fluxes U_m are defined using Darcy's law 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 m, K the permeability tensor, λ_m the mobility of a component, and P_m the pressure of a phase. Additional equations specifying volume, capillary, and state constraints are added, and boundary and initial conditions complement the system. Finally, $N_m = S_m \rho_m$ with S_m denoting saturation of the phase m (for more information, see e.g., [3]).

The continuous problem is discretized using mixed finite elements. The resulting discrete model is solved by the IPARS (Integrated Parallel Accurate Reservoir Simulator) software developed at the Center for Subsurface Modeling at The University of Texas [6]. IPARS is a parallel reservoir simulation framework for modeling multiphase, multiphysics flow in porous media [7,9].

2.2 The economic revenue 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. Neglecting fixed costs, we define our objective function by summing, 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:

$$f_T(p) = -\int_0^T \left\{ \sum_{prod.wells} \left[c_0 \left| q_o(s) \right| - c_{w,disp} \left| q_w(s) \right| \right] - \sum_{inj.wells} c_{w,inj} \left| q_w(s) \right| \right\} (1+r)^{-t} \, dt \,. \tag{2}$$

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 dollars per barrel each. The exponential factor models that up-front costs have to be paid off with interest. Finally, we define $f_T(p)$ to be the negative total revenue. With the simulation model and objective function described above, the optimization problem reads: *Find the optimal well location* p^* *such that*

$$p^* = \arg\min_{p \in P} f_T(p). \tag{3}$$

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. We note that in the discrete model, possible well locations are only cell-block mid-points. P is thus a bounded integer lattice.

2.3 Optimization algorithms

The first optimization algorithm we consider is an integer version of the Simultaneous Perturbation Stochastic Approximation (SPSA) method. First introduced by Spall [8], it uses the following idea: in each 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.

In its basic form, 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 in [5]. 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 $\lceil . \rceil$ 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 that does not modify points inside these bounds. Then the integer SPSA algorithm which we will use for the computations in this paper is stated as follows (for more details see [1,2]):

Algorithm 2.1 (Integer SPSA)

1. Set k=1, $\gamma = 0.101, \alpha = .602$
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 $c_k = \left\lceil \frac{c}{k^\gamma} \right\rceil$, $a_k = \frac{a}{k^\alpha}$.
   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 $g_k = (f^+ - f^-)/\left|\Pi(p_k + c_k\Delta_k) - \Pi(p_k - c_k\Delta_k)\right|$.
   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

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, 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. This algorithm requires $2n$ function evaluations per iteration, in contrast to the 2 evaluations in the SPSA method. However, we can expect better search directions from it.

3. The Grid computing paradigm

The Grid is rapidly emerging as the dominant paradigm for wide area distributed computing [4]. Its goal is to provide a service-oriented infrastructure that leverages standardized protocols and services to enable pervasive access to, and coordinated sharing of geographically distributed hardware, software, and information resources.

The software implementing the various steps in the computations outlined in the previous section consists of a number of modules that are coupled together using this Grid technology. These modules are (see Fig. 1): (1) The optimization service; (2) the IPARS factory, that upon request starts IPARS for a given well location or retrieves previous solutions from a database system; (3) IPARS itself, performing the parallel simulation; (4) the economic revenue model that takes the IPARS output and computes the economic value of the oil produced in the scenario for which the simulation was run; and, (5) the clients requesting the optimization service.

These modules can run on different machines at physically different locations, and communicate with each other as Grid enabled services using the DISCOVER Computational Collaboratory that provides the mechanisms for distributed applications [2]. In particular, DISCOVER provides the tools to find, interact with, and enquire the capabilities of services running on distributed machines coupled via the Internet. It builds on the Globus toolkit, CORBA Commodity Grid, and a number of other standardized services.

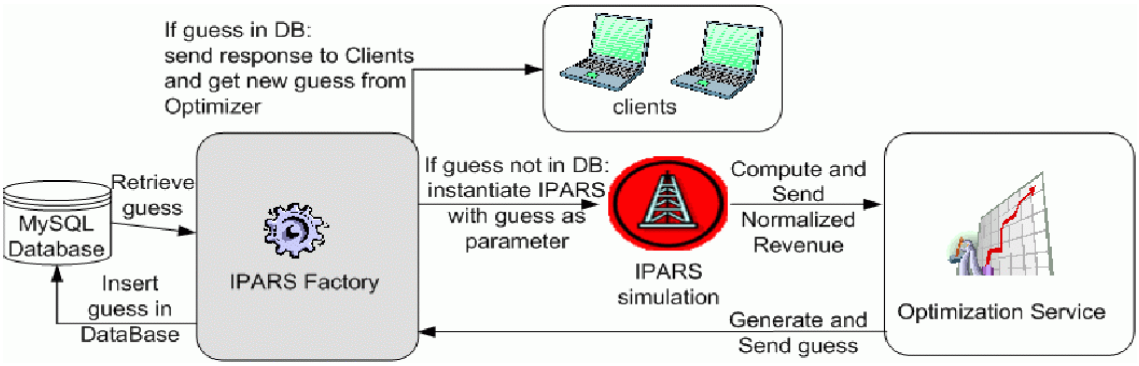

Figure 1. Optimization process on the Grid.

The advantages of this approach are apparent: while the optimization server can run on a user's desktop machine, it communicates with a central IPARS factory instance that can start IPARS runs on other machines, including on cluster computers and workstations, thus distributing the actual workload to available resources in the most efficient way [2].

4. Numerical results

We consider a relatively simple 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*. Hence, the model consists of 3904 grid blocks. The reservoir under study is located at a depth of 1 *km* and corresponds to a 2D section extracted from a real field in the Gulf of Mexico. The porosity has been fixed at $\phi = 0.2$ but the reservoir has a heterogeneous permeability field. It is assumed to be surrounded by impermeable rock. The fluids are initially in equilibrium with water pressures set to 2600 *psi* and oil saturation to 0.7. The reservoir model consists of 5 fixed wells: 2 water injectors at the bottom left and 3 oil producers at the top right corner of the domain. Injection and production rates are computed by specifying a fixed bottom hole pressure. The location of a third injection well is subject to optimization. The set of possible well locations in the discrete model is the integer lattice of grid block midpoints, i.e. $P = \{40,120,200,...,4840\} \times \{40,120,200,...,5080\}$.

In all cases, we ran our simulations for T=2000 days. We undertook to generate a large realistic data set for the evaluation of optimization algorithms by computing economic revenues for all 3904 possible well locations, and for 200 different time horizons T between 200 and 2000 days.

The data set therefore consists of $f_T(p)$ for 200 values of T and 3904 values for p. A plot of $f_{2000}(p)$ is shown in Fig. 3. Each simulation for a particular p took 20-30 minutes on a Linux PC with a 2GHz AMD Athlon processor, for a total of almost 2000 CPU hours.

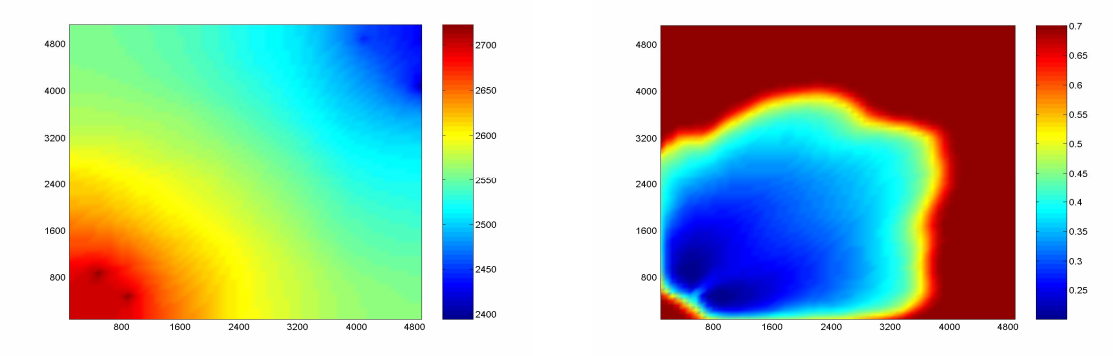

Figure 2. Pressure (left) and saturation (right) contours after 2000 days of simulation.

Note that while we would in general like 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 Fig. 3 has 72 local optima, with the global optimum being $f\left(p=(2920,920)^t\right)=-1.09804\cdot 10^8$. However, there are 5 more local extrema within only half a percent of this optimal value, which makes finding the global optimum rather complicated. The white marks in Fig. 3 (right) indicate the best well positions found by the SPSA algorithm when started from 7 different points. As can be seen, SPSA is able to find good locations from arbitrary starting points, even tough it is unable to find the global optimum every time. A typical optimization run showing both the path SPSA takes and convergence of the objective function is depicted in Fig. 4.

In order to evaluate the Integer SPSA algorithm, we start it from every possible location p_i in the set P, and for each of these 3904 SPSA 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. Since multiple function evaluations at the same point can be cached and are therefore inexpensive, we also record the number L_i of *unique* function evaluations.

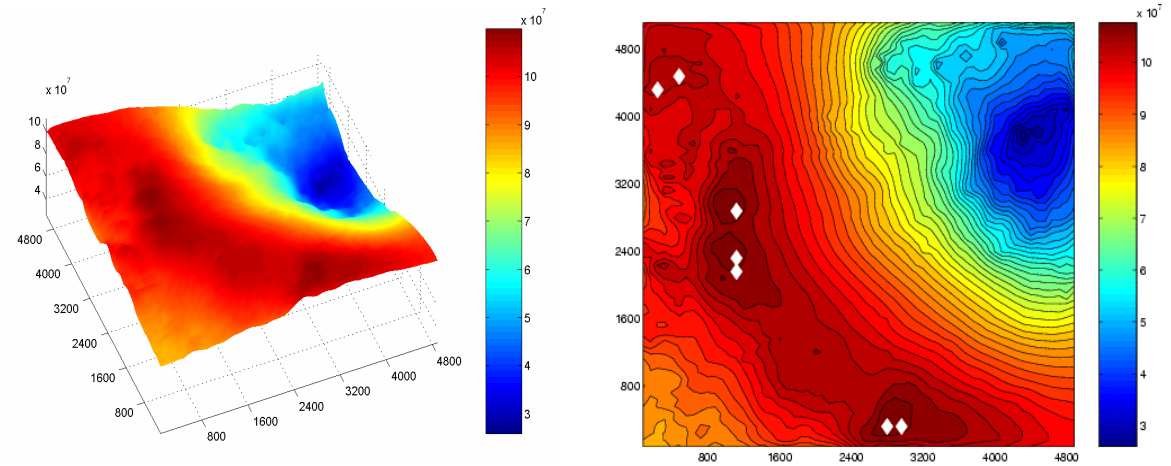

Figure 3. Surface view of the search space (left) and contour view with solutions obtained by SPSA with different initial guesses (right, with solutions indicated by white diamonds).

We consider the following statistical properties of SPSA: (1) What is the average value $\bar{f} = \sum_{i=1}^{3904} f(\hat{p}_i)/3904$, and how close is it to $f(p^*)$? (2) what is the value φ^{50} such that $f(\hat{p}_i) \geq \varphi^{50}$ in 50% of runs (and similar for the 95th percentile, φ^{95})? And, (3) what are the average number $\bar{K} = \sum_{i=1}^{3904} K_i / 3940$ and $\bar{L} = \sum_{i=1}^{3904} L_i / 3904$ of function evaluations and unique function evaluations to obtain convergence?

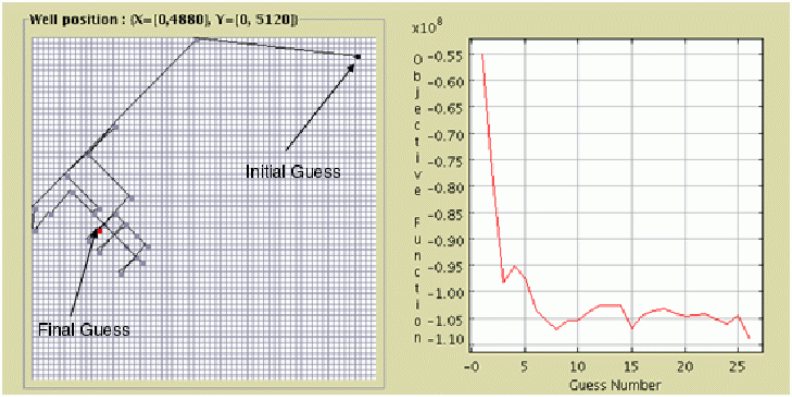

Figure 4. Computed well positions and corresponding revenue value during optimization.

The answer to the first two questions is closely related to the probability with which the algorithm terminates at any given point $p \in P$. For the two functions $f_{500}(p)$ and $f_{2000}(p)$, this probability of stopping at p is shown in Fig. 5. It is obvious that the locations where the algorithm stops are close to the (local) optima of the solution surface.

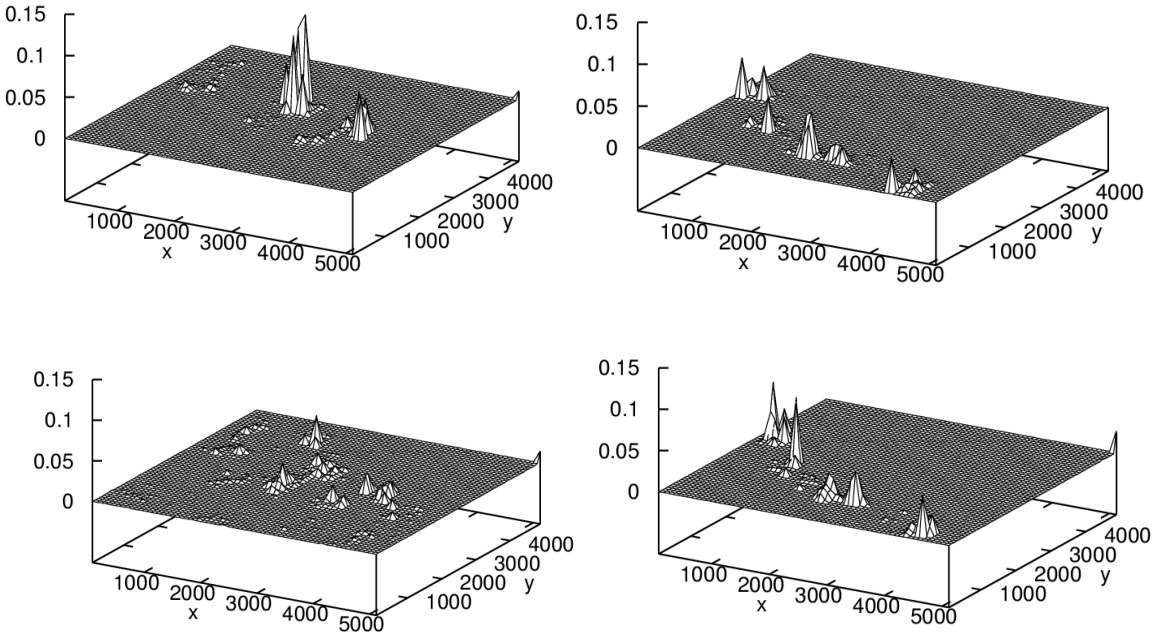

Figure 5. Probability surface for the SPSA algorithm at 500 days (top-left),2000 days (top-right) and for the FDG algorithm at 500 days (bottom-left) and 2000 days (bottom-right) of simulation.

T	$f_T(p^*)$	$\bar{f}_T$	φ_T^{50}	φ_T^{95}	$\overline{K}_T$	$\overline{L}_T$
500	$-2.960\cdot10^7$	$-2.853\cdot10^7$	$-2.920\cdot10^7$	$-2.507\cdot10^7$	52.2	42.6
1000	$-6.696\cdot10^7$	$-6.412\cdot10^7$	$-6.490\cdot10^7$	$-5.834\cdot10^7$	41.0	33.1
1500	$-9.225\cdot10^7$	$-9.011\cdot10^7$	$-9.139\cdot10^7$	$-8.286\cdot10^7$	40.8	33.1
2000	$-1.098\cdot10^8$	$-1.075\cdot10^8$	$-1.086\cdot10^8$	$-1.046\cdot10^8$	37.8	30.2

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}	$\overline{K}_T$	$\overline{L}_T$
500	$-2.960\cdot10^7$	$2.708\cdot10^7$	$-2.794\cdot10^7$	$-2.232\cdot10^7$	52.6	24.4
1000	$-6.696\cdot10^7$	$-6.222\cdot10^7$	$-6.480\cdot10^7$	$-5.572\cdot10^7$	53.0	28.1
1500	$-9.225\cdot10^7$	$-8.837\cdot10^7$	$-9.133\cdot10^7$	$-8.211\cdot10^7$	55.5	32.4
2000	$-1.098\cdot10^8$	$-1.062\cdot10^8$	$-1.083\cdot10^8$	$-1.044\cdot10^8$	57.0	31.5

Table 2. Statistical properties of termination points of the integer FDG algorithm.

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 how often to restart the optimization algorithm from different starting positions. Such restarts may be deemed necessary since 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 an improved initial guess. While φ^{95} 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. 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; whereas, if we cache previously computed values, only 30-42 function evaluations are required.

Table 2 show the results obtained for the Finite Difference Gradient algorithm. From the table, it is obvious that for earlier times, the algorithm needs less function evaluations. However, it also produces worse results, being significantly farther away from the global optimum on average. The reason for this is seen in the bottom row of Fig. 5 where we show the termination points of the algorithm: it is apparent that the algorithm quite frequently gets stuck in local optima, at least much more frequently than the SPSA algorithm. This leads to an early termination of the algorithm and suboptimal overall results.

5. Conclusions

We have presented the application of integer versions of the SPSA and the FDG optimization algorithms to find the optimal well location on a realistic dataset from the Gulf of Mexico. In order to validate their solutions, an exhaustive evaluation of the search space was performed. We have compared the quality of solutions as well as their efficiency. In particular, we have shown

that the SPSA algorithm yields very good solutions with relatively few function evaluations, and is capable of handling situations with many local extrema significantly better than a simple gradient-based algorithm. This knowledge is useful for tackling larger problems for which the exact optimum is unknown and impossible to compute. Given the size of the problem and the expensiveness of functions evaluations (20-30 minutes of CPU time per function evaluation), the computations were performed using a Grid enabled software framework, in which optimizers, simulators, and function evaluators are separate agents on different, geographically distributed machines. These agents are able to automatically detect the availability of computational resources, and start instances of the IPARS reservoir simulation software on appropriate machines while still providing the researcher with the ability to control the various distributed processes from the desktop. We presently prepare a more in depth comparative analysis of SPSA with other optimization algorithms for the well placement problem [1].

Acknowledgements
The research was supported by the National Science Foundation (NSF) via grants ACI 9984357 (CAREERS), EIA 0103674 (NGS), NSF EIA 0121523/EIA-0120934 (ITR), ANI-0335244 (NRT), CNS-0305495 (NGS) and by DOE ASCI/ASAP (Caltech) via grant 82-1052856.

References

[1] W. Bangerth, H. Klie, and M. F. Wheeler. *On the Reservoir Oil Placement Problem using the Simultaneous Perturbation Stochastic Approximation Method*, in preparation, 2004.

[2] W. Bangerth, H. Klie, V. Matossian, M. Parashar, and M. F. Wheeler. *An Automatic Reservoir Framework for the Stochastic Optimization of Well Placement*. Accepted for publication in Cluster Computing: The Journal of Networks, Software Tools and Applications, 2004.

[3] J. R. Fanchi, *Principles of Applied Reservoir Simulation*, Gulf Professional Publishing, 2nd. edition, 2001.

[4] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers, 1999.

[5] L. Gerencsér, S. D. Hill, and Z. Vágó. *Discrete optimization via SPSA*. In Proceedings of the American Control Conference, Arlington, 2001, pp 1503-1504, 2001.

[6] *IPARS: Integrated Parallel Reservoir Simulator*. http://www.ices.utexas.edu/CSM.

[7] Q. Lu, M. Peszynska, and M. F. Wheeler. *A parallel multi-block black-oil model in multi-model implementation*. SPE Journal, 7(3):278--287, September 2002. SPE 79535

[8] J. C. Spall. *Introduction to stochastic search and optimization: Estimation, simulation and control*. John Wiley & Sons, Inc., Publication, New Jersey, 2003.

[9] M. F. Wheeler, J. A. Wheeler, and M. Peszynska. *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, and G. F. Pinder, editors, Computational Methods in Water Resources, pp. 167-174. A. A. Balkema, 2000.