OPTIMIZATION OF FIELD DEVELOPMENT USING PARTICLE SWARM OPTIMIZATION AND NEW WELL PATTERN DESCRIPTIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ENERGY RESOURCES ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jérôme Emeka Onwunalu
June 2010

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Louis Durlofsky, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Roland Horne

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Tapan Mukerji

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

Abstract

The optimization of the type and location of new wells is an important issue in oil field development. Computational algorithms are often employed for this task. The problem is challenging, however, because of the many different well configurations (vertical, horizontal, deviated, multilateral, injector or producer) that must be evaluated during the optimization. The computational requirements are further increased when geological uncertainty is incorporated into the optimization procedure. In large-scale applications, involving hundreds of wells, the number of optimization variables and the size of the search space can be very large. In this work, we developed new procedures for well placement optimization using particle swarm optimization (PSO) as the underlying optimization algorithm. We first applied PSO to a variety of well placement optimization problems involving relatively few wells. Next, a new procedure for large-scale field development involving many wells was implemented. Finally, a metaoptimization procedure for determining optimal PSO parameters during the optimization was formulated and tested.

The particle swarm optimization is a population-based, global, stochastic optimization algorithm. The solutions in PSO, called particles, move in the search space based on a "velocity." The position and velocity of each particle are updated iteratively according to the objective function value for the particle and the position of the particle relative to other particles in its (algorithmic) neighborhood. The PSO algorithm was used to optimize well location and type in several problems of varying complexity including optimizations of a single producer over ten realizations of the reservoir model and optimizations involving nonconventional wells. For each problem, multiple optimization runs using both PSO and the widely used (binary) genetic algorithm (GA) were performed. The optimizations showed that, on average, PSO provides results that are superior to those using GA for the problems considered.

In order to treat large-scale optimizations involving significant numbers of wells, we next developed a new procedure, called well pattern optimization (WPO). WPO avoids some of the difficulties of standard approaches by considering repeated well patterns and then optimizing the parameters associated with the well pattern type and geometry. WPO consists of three components: well pattern description (WPD), well-by-well perturbation (WWP), and the core PSO algorithm. In WPD, solutions encode well pattern type (e.g., five-spot, seven-spot) and their associated pattern operators. These pattern operators define geometric transformations (e.g., stretching, rotation) applied to a base pattern element. The PSO algorithm was then used to optimize the parameters embedded within WPD. An important feature of WPD is that the number of optimization variables is independent of the well count and the number of wells is determined during the optimization. The WWP procedure optimizes local perturbations of the well locations determined from the WPD solution. This enables the optimized solution to account for local variations in reservoir properties. The overall WPO procedure was applied to several optimization problems and the results demonstrate the effectiveness of WPO in large-scale problems. In a limited comparison, WPO was shown to give better results than optimizations using a standard representation (concatenated well parameters).

In the final phase of this work, we applied a metaoptimization procedure which optimizes the parameters of the PSO algorithm during the optimization runs. Metaoptimization involves the use of two optimization algorithms, where the first algorithm optimizes the PSO parameters and the second algorithm uses the parameters in well placement optimizations. We applied the metaoptimization procedure to determine optimum PSO parameters for a set of four benchmark well placement optimization problems. These benchmark problems are relatively simple and involve only one or two vertical wells. The results obtained using metaoptimization for these cases are better than those obtained using PSO with default parameters. Next, we applied the optimized parameter values to two realistic optimization problems. In these problems, the PSO with optimized parameters provided comparable results to those of the default PSO. Finally, we applied the full metaoptimization procedure to realistic cases, and the results were shown to be an improvement over those achieved using either default parameters or parameters determined from benchmark problems.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my adviser Prof. Louis J. Durlofsky for his support, encouragement and guidance throughout my graduate study at Stanford University. His comments and suggestions were extremely valuable to my research.

I thank my committee members Prof. Michael Saunders, Prof. Roland Horne, Prof. Tapan Mukerji, and Prof. Juan L.F. Martinez for their useful comments regarding my research. I am grateful to Dr. David Echeverria for his suggestions regarding my research and also for our numerous discussions about optimization techniques. I am grateful to Huanquan Pan for help with GPRS and to Prof. Marco Thiele for providing the 3DSL simulator and the reservoir model used in one of the examples.

I am grateful to the SUPRI-HW, SUPRI-B and Smart Fields industrial affilliates programs for financial support throughout my graduate study. I also thank the Stanford Center for Computational Earth and Environmental Science (CEES) for providing the computing resources used in this work. Special thanks goes to Dennis Michael for providing help with the computing clusters and for allowing me to run many jobs on the clusters.

I thank the BP Reservoir Management Team, Houston, Texas, for the internship opportunities. The experience was very valuable to my research at Stanford. Special thanks goes to Mike Litvak, Subhash Thakur, Bob Gochnour and Karam Burn for their support.

I am grateful to all my friends and my family for their love and support over the years.

Chapter 1

Introduction and Literature Review

Field development optimization involves the determination of the optimum number, type, location, trajectory, well rates, and drilling schedule of new wells such that an objective function is maximized. Examples of objective functions considered include cumulative oil (or gas) produced and net present value (NPV). The optimization task is challenging, because many wells may be required and different well types (vertical, horizontal, deviated or multilateral; producer or injector) may have to be evaluated. The incorporation of geological uncertainty, treated by considering multiple realizations of the reservoir, further increases the complexity of the optimization problem.

The computational demands of these optimizations are substantial, as the objective function values of many field development scenarios must be computed. Each evaluation requires performing a simulation run, and for large or complicated reservoir models, the simulation run times can be large. The number of simulations required depends on the number of optimization variables, the size of the search space, and on the type of optimization algorithm employed.

In large-scale field development problems, the number of wells required can be substantial; up to several hundred wells in recent applications. This increases the complexity of the optimization problem. Furthermore, the performance of the underlying optimization algorithm may degrade for very large numbers of optimization variables. It is therefore essential to have efficient and robust optimization procedures for this family of optimization problems.

In this work, we evaluated the particle swarm optimization (PSO) algorithm for well placement optimization problems. Results using PSO were compared to those obtained using a binary genetic algorithm. Next, we developed a new procedure, called well pattern optimization (WPO), which can be used for optimization problems involving a large number of wells arranged (essentially) in patterns. Finally, we applied a metaoptimization procedure to improve the performance of the general PSO algorithm for field development optimization problems.

1.1 Literature Review

The literature related to well placement optimization is very extensive. Different optimization algorithms, hybrid techniques, surrogate models (or proxies), constraint handling methods, and applications, have been presented. In the next section, we review the well placement optimization literature in the aforementioned categories. Next, we discuss relevant PSO research including PSO parameter selection and metaoptimization techniques.

1.1.1 Well Placement Optimization

Optimization algorithms

The well placement optimization problem is a high-dimensional, multimodal (for nontrivial problems), constrained optimization problem. The optimization algorithms employed for this problem fall into two broad categories: global search, stochastic algorithms and gradient-based algorithms. The stochastic optimization algorithms, such as genetic algorithms (GAs) and simulated annealing, are computational models of natural or physical processes. They do not require the computation of derivatives. In addition, stochastic optimization algorithms possess mechanisms or algorithmic operators to escape from local optima, e.g., the mutation operator in GAs [1]. However, these algorithms tend to require many function evaluations and their performance depends on the tuning of algorithmic parameters [1, 2, 3].

Gradient-based optimization algorithms require the computation of gradients of the objective function. The gradients can be computed using adjoint procedures or by numerical finite differences. Gradient-based algorithms seek to improve the objective function value in each iteration by moving in an appropriate search direction. Thus, gradient-based algorithms are computationally efficient, though they are susceptible to getting trapped in local optima. In the following sections, we discuss the specific stochastic and gradient-based algorithms employed for well placement optimization.

Stochastic optimization algorithms

The most common stochastic optimization algorithms for well placement optimization are simulated annealing (SimA) and GA. The SimA algorithm uses the analogy of metal cooling to find solutions to optimization problems [4]. SimA starts with a point in the search space and evaluates the objective function at this point. A new point is generated by a small perturbation of the current solution. Next, the objective function value at this new point is evaluated, and if the function value is lower (for minimization), the new point is accepted as the new starting point. However, if the new point has a higher objective function value, it may be accepted with a probability proportional to a “temperature” parameter that determines the progress of the algorithm. The algorithm is run until the temperature reaches some minimum value.

In [5] the SimA algorithm was applied to maximize NPV by optimizing the schedule and location of horizontal wells with fixed orientations. The well placement optimization problem was first formulated as a traveling salesman problem (TSP) with potential well locations represented as cities on the TSP tour. The drilling schedule was determined by the sequence for visiting the cities. The resulting TSP was then solved by the SimA algorithm. The procedure was successfully applied to optimize the locations of 12 wells. However, the TSP formulation is not efficient for well placement optimization problems because, in practice, every feasible grid block is a potential well location. In that case, the TSP tour becomes large due to the many well locations to be evaluated. Furthermore, the TSP is a difficult optimization problem whose complexity increases with the number of tours [6, 7]. Other authors [8, 9] have also used SimA, but they applied the algorithm directly to optimize well locations; i.e., a TSP formulation was not used.

Another type of stochastic optimization algorithm applied for well placement optimization is the genetic algorithm (GA). GA appears to be the most popular optimization algorithm employed for well placement and other reservoir-management-related applications [10, 11, 12]. There have been many successful applications of GA for well placement optimization–see, for example [8, 13, 14, 15, 16, 17, 18, 19, 20, 21]. GA is a computational analog of the process of evolution via natural selection, where solutions compete to survive. GAs represent potential solutions to the optimization problem as individuals within a population. The fitness (solution quality) of the individual evolves as the algorithm iterates (i.e., proceeds from generation to generation). At the end of the simulation, the best individual (individual with highest fitness) represents the solution to the optimization problem. Simple GA uses three operators, selection, crossover, and mutation [22, 23, 24] to generate new individuals from existing individuals.

The two main variants of the GA are the binary GA (bGA) and the continuous or real-valued GA (cGA). In binary GA, the optimization parameters (e.g., i, j, k locations of well heel and toe) are encoded and manipulated as binary variables. Necessary conversions from binary to decimal are performed before the function evaluation step. Most previous GA implementations have involved bGA, though applications using cGA were recently presented [15, 25]. GA-based procedures have been applied to optimize the locations of both vertical wells [8, 14, 16, 17, 19, 26, 21, 27] and nonconventional wells [13, 18, 20, 28, 29].

Another type of evolutionary algorithm, the covariance matrix adaptation evolution strategy (CMAES), was recently applied to optimize nonconventional well locations [30]. The CMAES algorithm was found to provide comparable results to those obtained using a bGA.

The solutions obtained using GA can be improved by combining GA and other optimization algorithms, e.g., ant colony algorithm [31], Hooke-Jeeves pattern search algorithm [20], polytope algorithm [14, 17, 21, 26] or tabu search [14]. These hybrid algorithms have been demonstrated to provide better results and reduce computational expense compared to using only GA [20, 31, 32].

Gradient-based optimization algorithms

Gradient-based optimization algorithms that have been applied for the optimization of well location include stochastic approximation and adjoint-based gradient algorithms. In [8], the simultaneous perturbation stochastic approximation (SPSA) algorithm was applied to optimize the location of vertical wells. The SPSA algorithm [33] is an approximate gradient-based algorithm. To compute the gradient, a random direction is first generated at the current point. The random direction is used to generate two new points and function evaluations are performed at these new points. Using the two function evaluations, the direction of increase (for maximization) or decrease (for minimization) in the objective function can be determined [8, 33]. The benefit of the SPSA algorithm is that the computation of gradients is independent of the number of variables in the optimization problem, as only two function evaluations are required to approximate the gradient. In [8] a finite difference gradient algorithm (FDG) was also applied to optimize well locations. The FGD algorithm is similar to SPSA, except that in the former, the gradients are computed using two-sided finite difference approximations for each optimization variable. While the gradients computed using the FDG procedure are comparably more accurate [8, 33], the number of function evaluations required to provide the gradient is much larger than in SPSA. The SPSA algorithm was reported to be better than the FDG algorithm for optimizing well locations. In addition, the FDG algorithm was found to be more susceptible to getting stuck in local optima [8].

In [8], the SPSA algorithm was compared to bGA, very fast SimA (VFSA), and Nelder-Mead simplex algorithms. The SPSA algorithm was found to perform better than the other algorithms in optimizations involving vertical wells. However, the use of the SPSA algorithm presents some challenges. The step size for calculating new solutions must be chosen carefully, otherwise new solutions may be infeasible. More generally, the objective function surfaces for well placement optimization problems can be very discontinuous, especially when the permeability field is highly heterogeneous. In such situations, gradients are not well defined and gradient-based algorithms may encounter problems.

Other gradient-based algorithms have also been applied for well placement optimization. In [34] a constrained quasi-Newton method was applied to optimize vertical well locations by minimizing pressure drawdown. Pressure drawdown was computed semi-analytically, while the gradients of pressure drawdown with respect to the well locations were computed numerically. In [35], the problem of optimizing the number and location of injection wells in a two-dimensional reservoir model was considered. A steepest descent algorithm was employed for the optimization. In [36], an adjoint method for well placement optimization in two-dimensional models was presented. These authors placed "pseudowells," producing or injecting at low rates, at each of the eight blocks surrounding the current well location. At each iteration, the adjoint method was used to compute rate gradients for each of the pseudowells at each time step. The gradients at the pseudowells were then summed, and the well was moved in the direction of the pseudowell with the largest summed gradient. The gradients computed in [36] are not with respect to the original well location, but with respect to the rates at the pseudowells. A similar pseudowell technique was applied in [37], though in this method the gradient of the objective function was computed with respect to continuous well locations. This approach allows for arbitrary search directions and step sizes.

The adjoint-based methods have the advantage of high computational efficiency. As is the case with all gradient-based algorithms, however, they are susceptible to getting trapped in local optima, so the optimized solutions will depend on the starting points. In addition, the use of pseudowells can pose challenges in cases with many wells or with non-conventional wells such as deviated or multilateral wells (no such wells were considered in [36, 37]), although a modified pseudowell technique for well trajectory optimization was recently presented in [38]. Finally, adjoint methods require access to the simulation code, which is not a requirement for any of the other methods considered above.

Well placement optimization with stochastic optimization algorithms requires a large number of function evaluations, each of which entails a simulation run. The computational expense can be reduced in several ways, including performing the simulations in parallel on distributed processors, using coarse reservoir models [25], semianalytic modeling [34], and the use of surrogate models (proxies). In the next section, we describe some of the proxies used in well placement optimization.

Use of proxies

Proxies are computationally fast but approximate models which are incorporated into optimization procedures. They reduce computational demands by reducing the number of full simulations performed during the optimization. Proxies can provide estimates of the objective function value of new development scenarios using previously simulated scenarios. The estimated objective function values can then be used to select promising scenarios for simulation during the optimization.

Examples of proxies used in well placement optimization include kriging [17, 21, 26, 39], least squares [39], neural networks [17, 20, 40], cluster-based statistical proxies [13, 41], and neuro-fuzzy methods [19, 42]. Other authors have used proxies based on reservoir parameters to screen well locations, e.g., productivity index [30], productivity potential [43], and quality maps of oil and gas produced [19]. In [19, 30, 43], the objective of the optimization was to place the wells in reservoir locations that maximized the screening parameters.

Field development constraints and treatment

In well placement optimization, two kinds of constraints commonly considered are bound and practical constraints. Bound constraints on the variables arise because solutions are sought within specified variable ranges. For example, all wells must be drilled in the feasible reservoir region. Practical constraints are related to the field development project, and examples include [11, 29]: well-pair distance constraints, drilling schedule constraints, well rate constraints, production facility constraints, constraints on the number of wells, etc. The incorporation of these constraints increases the difficulty of the optimization problem [3, 6].

Different approaches have been employed for handling infeasible development scenarios in well placement optimization. The most common method is the penalty method where infeasible solutions are penalized [17] or assigned a large negative NPV [44, 45]. Other investigators handle constraints using different solution encoding or specialized algorithms. In an application involving irregular reservoir boundaries, GA individuals were encoded using a one-dimensional list of the feasible grid blocks [14]. A procedure for handling different well placement optimization constraints such as maximum well length and minimum distance between wells was presented in [29]. These investigators used a two-population binary GA where each individual belongs to one of the two populations depending on its feasibility. When an infeasible individual is encountered, a special operator is used to "repair" the individual until it becomes feasible.

1.1.2 Large-Scale Field Development Optimization

Large-scale field development optimization problems involve optimizing the location and type of a large number of wells, with recent applications involving several hundred wells [44, 45]. A straightforward approach for representing the solution parameters in such cases is to consider a series of wells and to concatenate the well-by-well optimization parameters. For problems with many wells, however, the number of optimization variables becomes large, thereby increasing the complexity of the optimization problem. The performance of the underlying optimization algorithm many degrade for very large numbers of variables. For example, if a bGA is employed for the optimization of hundreds of wells, very long chromosomes will result. Because the population size in GA is determined from the length of the chromosome (e.g., it can be chosen to be equal to the number of bits in the chromosome [17, 22]), large population sizes will be required to achieve acceptable algorithm performance. This in turn leads to high computational expense. Additional complications may result when necessary constraints (e.g., minimum well-to-well distances) are incorporated, and this can negatively impact algorithm performance.

One way to approach large-scale field development optimization problems is to consider alternative solution representation techniques, which lead to a reduction in the size of the search space or number of optimization variables. Different solution representation techniques have been considered. In [14], a bGA was used to optimize the locations of vertical and horizontal wells (with fixed orientation) in a reservoir with noncommunicating feasible regions and irregular boundaries. In these optimizations, feasible well locations were represented by a one-dimensional vector. This technique prevents wells from being located outside the feasible reservoir regions and was successfully applied to optimize the location of 33 new wells in a real field application. However, the length of the GA individual will increase with the number of wells considered. It was shown in [16] that this approach is not efficient because of major discontinuities in the search.

Another solution representation technique that leads to reduction in the number of variables is to consider well patterns in the optimization. Kriging and least-square algorithms were applied in [39] to optimize the location of an inverted five-spot pattern element in a waterflooding project. The pattern element was represented with four optimization variables: the spatial location of the injector and two well spacing parameters. A fixed pattern approach (FPA) for optimizing wells in reservoirs with irregular boundaries was also used in [32]. In FPA, wells are optimized in a line drive pattern using two optimization variables - well spacing and distance to the reservoir boundary. The procedure, which was applied to a field case, reduced the number of simulations required in the optimization. A robust field development procedure, described in [45], was applied successfully to giant fields [44, 46]. To reduce the number of optimization variables, three different pattern types were considered: inverted five-spot, inverted seven-spot, and staggered line drives. These investigators considered different well types and a specified set of well spacings in their optimizations.

In optimizations using concatenated well-by-well parameters, treatment of well-pair distance constraints requires specifying user-defined threshold values. These values have the effect of reducing the search space of solutions considered and may affect the quality of solutions obtained. In optimizations with well patterns, the optimum pattern type, size, and orientation for a given application are unknown. Previous applications with well patterns consider a single pattern type [32, 39], or a set of well patterns with fixed well spacings [45]. In this thesis, we introduce a new well pattern optimization procedure that generalizes previous techiniques. We will show that this approach is well suited for optimizing large-scale field development.

1.1.3 Particle Swarm Optimization (PSO) Algorithm

The particle swarm optimization (PSO) algorithm [47, 48, 49, 50] is a relatively new algorithm for global optimization. The algorithm mimics the social interactions exhibited in animal groups, e.g., in fish swarms and in bird flocks. Like the GA, PSO is a population-based algorithm. PSO solutions are referred to as particles rather than individuals as in GA. The collection of particles in a given iteration is called the swarm. The position of each particle is adjusted according to its fitness and position relative to the other particles in the swarm.

The GA and PSO algorithms share some similarities. Both algorithms have operators to create new solutions from existing solutions. Both also include random components to prevent solutions from being trapped in local optima. The algorithms differ, however, in the number and type of operators used to create and improve solutions during the optimization. GA has three main operators: selection, crossover, and mutation. There are many strategies for applying these operators [51, 52], and the best option will depend on the specific optimization problem [7, 22]. The basic PSO algorithm, by constrast, has one main operator, the "velocity" equation, which consists of several components and moves the particle through the search space with a velocity (though, in PSO, each particle also carries a memory). The velocity provides the search directions for each particle, and is updated in each iteration of the algorithm. The GA and PSO algorithms also differ in the number of vectors associated with each individual or particle. In GA, there is one solution vector for each individual. However, for PSO, there are three vectors associated with each particle: current position, velocity, and previous best position.

The PSO algorithm uses a cooperative search strategy for optimization where particles interact with each other. This interaction is achieved using neighborhoods, where a particle can only interact with other particles in its neighborhood [2, 3, 53]. Depending on the number of neighborhoods used, the global best (gbest) and local best (lbest) PSO variants are obtained [3]. In gbest PSO, a single neighborhood containing all the particles is used. For the lbest PSO, more than one neighborhood is employed in the optimization, and each particle may belong to multiple neighborhoods.

The computation of particle velocities at each iteration uses the locations of the best particles found so far. Particle velocities are computed similarly for the gbest and lbest PSO variants, except that in gbest PSO, the best particle in the entire swarm is used, while in the lbest PSO, the best particle in a particle's neighborhood is used. The choice of neighborhood topology also affects the PSO algorithm performance and different types of neighborhood topologies (e.g., random, ring) have been developed [2, 3, 54, 55]. Faster convergence is observed for gbest PSO, but there is a higher susceptibility of getting trapped in local optima. On the other hand, the lbest PSO is slower, but can provide robust results especially in problems with many local optima [3, 53].

The PSO algorithm has been applied successfully in many different application areas such as training neural networks [47, 48, 56], dynamic economic dispatch problems [57], pole shape optimization [58], water reservoir operations and planning [59], placement of sensors for civil structures [60], geophysical inverse problems [61] and flow shop scheduling problems [62]. Although the PSO algorithm does not appear to have been applied previously within the context of oil reservoir simulation, it has been used for related subsurface flow applications. Specifically, in [63] PSO was applied to a contaminated groundwater remediation problem using analytical element models. The investigators minimized the cost of remediation by optimizing the number, location, and rates of (vertical) extraction and injection wells. Several algorithms were applied, including continuous GA, simulated annealing, and Fletcher-Reeves conjugate gradient. The best results were obtained using the PSO algorithm. The authors also compared the effectiveness of GA and PSO algorithms for the elimination of wells when the number of wells required was overspecified. The PSO algorithm was also found to be more effective for this application. These findings provide a key motivation for our work on applying the PSO algorithm to well placement optimization problems. Furthermore, the PSO algorithm has been found to provide better results, and in general to require fewer function evaluations, than GA [61, 62, 64] and SimA [61] algorithms for applications involving scheduling, geological inversion and computer hardware design.

Like other stochastic optimization algorithms, the performance of PSO depends on the values assigned to the parameters in the algorithm. We now discuss previous work related to choosing PSO parameter values.

PSO parameter selection

The PSO algorithm has several parameters which must be specified before performing an optimization run. These include population size, the maximum number of iterations, and the weights of the inertia (ω), cognitive (c_1), and social (c_2) components of the velocity equation [3]. These weights affect the trajectories of the particles. If a local best PSO variant is used, a neighborhood topology must also be specified. The swarm size affects the search ability of the PSO algorithm and it is chosen based on the size of the search space and problem difficulty [53]. Population sizes in the range of 20-40 were recommended in [2, 53].

Several authors have proposed specific PSO parameter values or techniques for obtaining appropriate parameter values. Values of $\omega = 1$, $c_1 = c_2 = 2.0$ were used in [47, 48]. These were heuristic values and have since been found to violate PSO convergence requirements [3, 53]. It has been shown that particle trajectories can be converging, cyclic or diverging [3]. Modifications have been introduced to curb particle divergence issues, e.g., use of velocity clamping (or restriction) and use of inertial weights other than unity, for example $\omega = 0.729$ in [65]. These modifications lead to better convergence characteristics of the PSO algorithm.

Others authors studied the stability and convergence of the PSO particle trajectories (see for example [50, 58, 66, 67, 68]) in order to understand particle dynamics and to choose optimal PSO parameters. These studies provided detailed mathematical analyses describing the relationships between particle trajectory and the values of $\omega,\ c_1$, and c_2. The different parameter regions (i.e., values of ω, c_1, and c_2) where a particle's trajectory would converge were shown in [50]. In [66], so-called parameter clouds were proposed where the values of $\omega,\ c_1$, and c_2 were selected based on the results of exhaustive sampling of $\omega,\ c_1$, and c_2 in optimizations involving mathematical functions, e.g., the Rosenbrock and Griewank functions. The first step in the cloud method is to select several parameter points (combinations of $\omega,\ c_1$ and c_2) from the regions that resulted in low objective function values (for minimization). Then, the selected parameters were used in other optimization problems. A constriction factor, which damps the computed particle velocity and ensures that the swarm converges, was introduced in [67]. Different PSO variants with parameter values determined from analysis of particle trajectories were developed in [68].

The work [61, 66, 67, 68] on particle swarm stability and convergence often involves some simplifications [3]. For example, these studies involve a single particle, and the effect of particle-particle interactions are not considered. Another method for determining optimal PSO parameters is to actually optimize the parameter values during optimization. This method is described in the next section.

1.1.4 Metaoptimization for Parameter Determination

The parameters in the PSO algorithm can be directly optimized for a given optimization problem [2, 69, 70, 71]. This method, referred to as metaoptimization [69, 70, 71], eliminates the need to specify parameter values for a given problem (the cloud method described in the previous section also eliminates this need).

Metaoptimization involves the use of two optimization algorithms. Within the context of PSO, the first algorithm optimizes the PSO parameters, while the second one optimizes the specific optimization problem using the PSO parameters obtained from the first algorithm. In practice, any optimization algorithm can be used for optimizing the PSO parameters. PSO is commonly employed for this purpose [2, 69, 70], although a continuous GA was used in [71].

We focus on applications that use PSO for optimizing the parameter values. The first and second PSO algorithms are referred to as the "superswarm PSO" and "subswarm PSO" respectively [69]. Each superswarm PSO particle corresponds to a point in the search space of PSO parameters (e.g., ω, c_1, c_2). The fitness of a superswarm particle is computed by performing several subswarm optimizations using the parameters from the superswarm particle. The subswarm optimizations are performed on either the target problem or one or more representative optimization problems, in our case, a well placement optimization. The fitness functions are defined differently for the subswarm and superswarm and depend on the objective function in the subswarm. In the subswarm, the objective is to minimize some error (if the global optimum is known) or maximize some objective function, e.g., NPV in well placement optimization problems. In the superswarm optimizations, the objective is to minimize average error, or maximize the average objective function value from several subswarm optimization runs. Multiple subswarm optimization runs are performed for each superswarm particle because of the stochastic nature of the PSO algorithm. The above-mentioned metaoptimization applications (except [2]) used a gbest PSO with a single neighborhood, while [2] considered two types of neighborhood topologies.

The metaoptimization procedure has been demonstrated to provide better results compared to standard PSO with unoptimized parameter values [69, 70, 71]. However, this method is computationally intensive due to the large number of function evaluations required. A large number of function evaluations is needed because the actual optimization problem is solved many times (in subswarm optimization runs) using different PSO parameters. As a result, many metaoptimization studies [2, 70, 71] have used computationally inexpensive mathematical test functions, e.g., Rosenbrock, Rastrigin and Sphere functions [2, 3], for the subswarm optimizations.

The metaoptimization procedure can be used in two ways. First, it can be applied to determine the best PSO parameters for a given set of small, benchmark optimization problems, where the objective function surfaces are known through exhaustive sampling. Then, the optimized PSO parameters are used to solve realistic problems in the hope that the PSO parameters are optimal (or at least reasonable) for these problems. In [2], several PSO parameters including population size, number of particles informed, topology type (random and circular), ω, c_1, and c_2, were optimized over several mathematical test functions. After the metaoptimizations, the number of particles informed (for most functions) was found to be 3. In all test functions considered, a variable random neighborhood topology was found to be better than a fixed circular topology.

Metaoptimization can also be applied directly to each new optimization problem. Such an application may be preferable because the best parameter settings will in general depend on the specific problem considered. In [69], a metaoptimization procedure was used for training neural networks, where PSO parameters, in addition to the number of hidden layers in the network, were optimized for the subswarm. These researchers reduced the computational expense by using a smaller number of particles and fewer iterations in the superswarm PSO. They also performed several PSO optimizations using the optimized parameters for similar optimization problems. The results were compared to those obtained using standard PSO (with parameters from the literature). The PSO with the optimized parameters was found to provide the best results. Specifically, [69] reported that the PSO with optimized parameters produced more robust results and converged faster for the training of neural networks. Motivated by these findings, we will explore the use of a metaoptimization procedure for determining PSO parameter values for well placement optimization problems.

1.2 Scope of Work

Optimizing the placement of new wells in a field development project is essential in order to maximize project profitability. This dissertation focuses on the development of efficient optimization algorithms and procedures for these types of optimizations. We applied the PSO algorithm to different optimization problems. We also devised a new procedure for large-scale field development optimization. In this methodology, the number of optimization variables does not increase with well count. Finally, we studied the use of metaoptimization techniques to improve the performance of the PSO algorithm for well placement optimization.

This objectives of this research were:

- to evaluate and apply the PSO algorithm for well placement optimization problems. We considered several optimization problems and compared PSO optimization results to those obtained using bGA. The examples considered involved different numbers of wells, well types (producer and injector, vertical and nonconventional), number of realizations, and size of the search space.
- to develop and apply new approaches for large-scale field development optimization involving many wells. We developed a new well pattern optimization (WPO) algorithm which contains two optimization phases. In the first phase, optimizations

were performed using a new (generic) well pattern description. In the (optional) second phase, phase 1 solutions were improved further using well-by-well perturbation. In both phases, we used the PSO algorithm for the underlying optimizations. We considered different optimization problems including a case with multiple reservoir models and a reservoir with an irregular boundary.

- to improve the performance of the PSO algorithm for well placement optimization using metaoptimization. For this study, we applied PSO metaoptimization techniques to a variety of well placement optimization problems.

1.3 Dissertation Outline

This dissertation is organized as follows. In Chapter 2, we discuss the application of the PSO algorithm to several well placement optimization problems. First, we describe the PSO algorithm in detail, presenting different variants of the algorithm, neighborhood topologies, and treatment of infeasible particles. We then consider several optimization problems of varying complexity in terms of the size of the search space, the dimension and size of the reservoir, and the number and type of wells considered. We compare PSO optimization results to those obtained with a binary GA implemenation. For all examples, we performed multiple optimization runs because of the stochastic nature of the GA and PSO algorithms.

The PSO algorithm performed very well for all of the optimization problems considered. In one example, we assessed the sensitivity of PSO and GA results to varying swarm (population) sizes. For small swarm/population sizes, the PSO algorithm achieved better results than bGA. The performance of both algorithms was about the same for large population sizes. The senstivity results indicated that PSO finds similar or better solutions than bGA using fewer function evaluations. These findings are in agreement with those observed by [61, 62, 64]. Other examples were also considered including optimizing the well type and location of 20 vertical wells, optimizing the location of four deviated producers, and optimizing the location of two dual-lateral wells. In these examples, PSO achieved better results (on average) than bGA. The work presented in Chapter 2 has been published in *Computational Geosciences* [72].

Chapter 2 describes the application of the PSO algorithm to a problem involving up to 20 vertical wells, in which the well-by-well optimization parameters are simply concatenated. Using this approach in large-scale field development projects would result in a great number number of variables and a very large search space. In Chapter 3, we introduce a new procedure, called the well pattern optimization (WPO) algorithm, which can be used for optimization problems involving a large number of wells. WPO consists of a new well pattern description (WPD), followed by an optional well-by-well perturbation (WWP), with both procedures incorporated into a core PSO methodology. WPD represents solutions at the level of well patterns rather than individual wells, which can lead to a significant reduction in the number of optimization variables. Using WPD, the number of optimization variables is independent of the number of wells considered. In WPD, each potential solution consists of three elements: parameters that define the basic well pattern, parameters that define so-called well pattern operators, and the sequence of application of these operators. The well pattern operators define pattern transformations that vary the size, shape and orientation of the well patterns considered in the optimization. The optimum number of wells required, in addition to the producer-injector ratio, is obtained from the optimization. Optimized solutions based on WPD are always repeated patterns; i.e., the method does not lead to irregular well placements. The subsequent use of WWP allows (limited) local shifting of all wells in the model, which enables the optimization to account for local variations in reservoir properties.

The WPO optimization procedure introduced here differs from the work of [45] in several respects. We used a different technique to represent potential field development scenarios, and our algorithm considers very general well patterns. This was accomplished through use of pattern transformation operations, which allow patterns to be rotated, stretched or sheared to an optimal degree. This can be important, for example, in situations where there are preferential flow directions in the field. In addition to standard patterns, the algorithm accepts user-defined patterns. WPD also eliminates the need for well-to-well distance constraints in the optimization. This is useful as highly constrained optimization problems are generally more difficult to solve than less constrained problems [3, 6]. Finally, the use of

WWP enables the local adjustment of well locations.

The WPD and WWP procedures were applied to different well placement optimization problems. In these examples, we performed multiple optimizations using the WPO procedure to gauge the degree of variability in the runs. In one problem, we compared WPD optimizations using one and four well pattern operators. The results show that better results were obtained using four well pattern operators. We performed both WPD and WWP optimizations for two examples, one involving five realizations of a reservoir model, and the other a reservoir with an irregular boundary. These optimizations demonstrated our ability to optimize large-scale field development. In the examples that use WWP optimizations, WWP was shown to improve the objective function value in each optimization run. Specifically, average improvements in NPV of 22% and 34% over the best WPD solutions were obtained.

The main benefits of the WPO procedure are that the optimum number of wells is determined during the optimization and that well-pair distance constraints do not need to be considered. Part of the work presented in Chapter 3 has been published in [73].

In the work described in Chapters 2 and 3, we applied PSO default parameter values suggested in the literature. These values were obtained from an analysis of particle trajectories and numerical experiments with mathematical functions [68]. In Chapter 4, we show the application of the metaoptimization procedure to optimize PSO parameters.

We first applied the metaoptimization procedure for four benchmark well placement optimization problems. Because of the very large number of function evaluations required by the metaoptimization procedure, we computed the full objective function surface exhaustively for each problem. This allowed inexpensive repetitions of optimization runs because we only needed to look up the NPV values. We demonstrated that the metaoptimization procedure provides better results than those obtained using PSO with unoptimized parameters. These results are consistent with those found in other PSO metaoptimization studies [68, 69].

Next, we used the metatoptimization procedure for realistic well placement optimization problems. We considered two optimization problems. In the first, we optimized the type and location of 15 wells using the well-by-well concatenation approach. In the second problem, we performed WPO optimizations. The metaoptimization results were shown to give better results than those using default parameters or parameters from the benchmark problems.

In Chapter 5, we present conclusions and recommendations for future research on the development of robust and efficient procedures for well placement optimization.

Chapter 2

Use of PSO for Well Placement Optimization

In this chapter, we describe the details of the PSO algorithm used in this work. Well placement optimization problems of varying complexity were considered. These problems included optimizing the location of a single producer over ten reservoir models, optimizing the location and type of 20 vertical wells, optimizing the location of four deviated producers, and optimizing the location of two dual-lateral producers. In each problem, we performed multiple optimizations using the PSO algorithm and compared results to those obtained using bGA. These results demonstrate the superior performance of PSO for the cases considered.

2.1 Particle Swarm Optimization (PSO) Algorithm

The PSO algorithm is a population-based stochastic optimization procedure developed by [47, 48]. The algorithm mimics the social behaviors exhibited by swarms of animals. In the PSO algorithm, a point in the search space (i.e., a possible solution) is called a particle. The collection of particles in a given iteration is referred to as the swarm. The terms ‘particle’ and ‘swarm’ are analogous to ‘individual’ and ‘population’ used in evolutionary algorithms such as GAs. We will use these terms interchangeably in this chapter.

At each iteration, each particle in the swarm moves to a new position in the search space. We denote $\mathbf{x}$ as a potential solution in the search space of a d-dimensional optimization problem, $\mathbf{x}_i(k) = \{x_{i,1}(k), \ldots, x_{i,d}(k)\}$ as the position of the ith particle in iteration k, $\mathbf{x}_i^{pbest}(k)$ as the previous best solution found by the ith particle up to iteration k, and $\mathbf{x}_i^{nbest}(k)$ as the position of the best particle in the neighborhood of particle $\mathbf{x}_i$ up to iteration k. We will discuss neighborhood topologies in detail in Section 2.1.1. One option is for the neighborhood to include the full swarm of particles, in which case, $\mathbf{x}_i^{nbest}(k) = \mathbf{x}^g(k)$, where $\mathbf{x}^g(k)$ is the global best particle position.

The new position of particle i in iteration $k+1$, $\mathbf{x}_i(k+1)$, is computed by adding a velocity, $\mathbf{v}_i(k+1)$, to the current position $\mathbf{x}_i(k)$ [47, 48, 65]:

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{v}_i(k+1) \cdot \Delta t, \tag{2.1}$$

where $\mathbf{v}_i(k+1) = \{v_{i,1}(k+1), \ldots, v_{i,d}(k+1)\}$ is the velocity of particle i at iteration $k+1$, and Δt is a 'time' increment. Here, consistent with standard PSO implementations, we set $\Delta t = 1$. It should be noted, however, that recent work has demonstrated improved results using variable Δt [50, 49], so this might be worthwhile to consider in future investigations. The elements of the velocity vector are computed as [3, 65]:

$$\begin{aligned} \mathbf{v}_i(k+1) = {} & \omega \cdot \mathbf{v}_i(k) \\ & + \; c_1 \cdot \mathbf{D}_1(k) \cdot (\mathbf{x}_i^{pbest}(k) - \mathbf{x}_i(k)) \\ & + \; c_2 \cdot \mathbf{D}_2(k) \cdot (\mathbf{x}_i^{nbest}(k) - \mathbf{x}_i(k)), \end{aligned} \tag{2.2}$$

where ω, c_1 and c_2 are weights; $\mathbf{D}_1(k)$ and $\mathbf{D}_2(k)$ are diagonal matrices whose diagonal components are uniformly distributed random variables in the range [0, 1]; and j, $j \in \{1, 2, \ldots, d\}$, refers to the jth optimization variable. In the optimizations performed in this chapter, we set $\omega = 0.721$ and $c_1 = c_2 = 1.193$. These values were determined from numerical experiments performed by [68]. We note that it is possible to optimize these parameters as part of the overall procedure. A metaoptimization procedure that optimizes the PSO parameters during optimization will be described in Chapter 4.

The velocity equation (Eq. 2.2) has three components, referred to as the inertia (term involving ω), cognitive (term involving c_1), and social (term involving c_2) components respectively [3]. The inertia component provides a degree of continuity in particle velocity from one iteration to the next, while the cognitive component causes the particle to move towards its own previous best position. The social component, by contrast, moves the particle toward the best particle in its neighborhood. These three components perform different roles in the optimization. The inertia component enables a broad exploration of the search space, while the cognitive and social components narrow the search toward the promising solutions found up to the current iteration.

Figure 2.1 shows the velocity computation and solution update in iteration $k+1$, for a particle in a two-dimensional search space. Here $\mathbf{v}_i(k)$ is the particle's previous velocity, while $\mathbf{v}_i^c(k)$ is the velocity (cognitive) from the current position ($\mathbf{x}_i(k)$) to the particle's previous best position ($\mathbf{x}_i^{pbest}(k)$), and $\mathbf{v}_i^s(k)$ is the velocity (social) from the current position to the current neighborhood best position ($\mathbf{x}_i^{nbest}(k)$). The velocity vectors $\mathbf{v}_i(k)$, $\mathbf{v}_i^s(k)$, and $\mathbf{v}_i^c(k)$ are used to compute $\mathbf{v}_i(k+1)$ according to Eq. 2.2. The new particle velocity, $\mathbf{v}_i(k+1)$, is added to the current position to obtain the new position vector, $\mathbf{x}_i(k+1)$, as shown in Eq. 2.1.

2.1.1 PSO Neighborhood Topology

Particle topologies or neighborhoods refer to the grouping of particles into subgroups. A particle can communicate and exchange information about the search space only with other particles in its neighborhood [2]. The performance of the PSO algorithm depends to some extent on the neighborhood topology, as discussed below. A particle j is in the neighborhood of particle i if there is a 'link' from particle i to j. This means that particle j informs particle i about its position in the search space. Particle j is called the informing particle (informant), while particle i is called the informed particle [2]. Each particle is a member of its neighborhood, i.e., each particle informs itself. The neighborhood size refers to the number of particles in the neighborhood.

The neighborhood topology is defined by a so-called adjacency matrix m_{ij}, where the rows correspond to informing particles and the columns correspond to the informed particles. In general, the matrix is nonsymmetric and contains zeros and ones as entries, with an entry of one indicating that particle i is contained in the neighborhood of particle j (particle i informs particle j). The matrix always has ones on the diagonal because each particle is

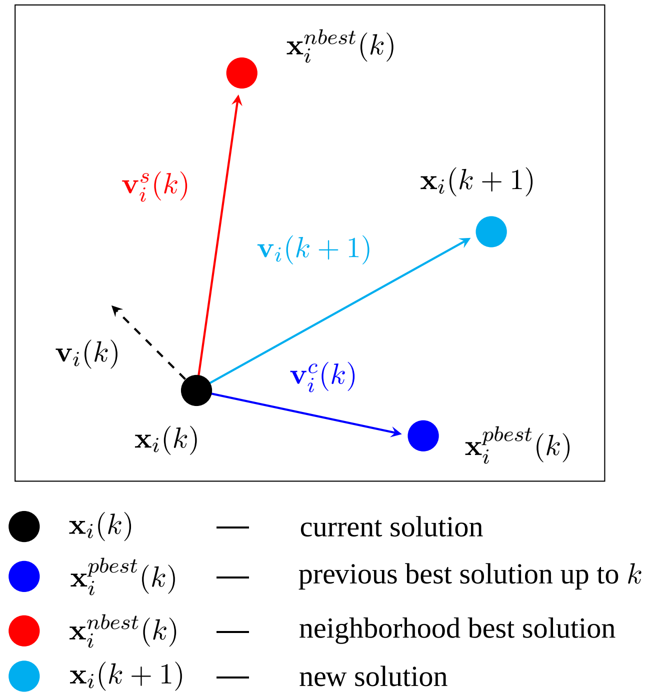

$\mathbf{x}_i^{nbest}(k)$
$\mathbf{v}_i^{s}(k)$
$\mathbf{x}_i(k+1)$
$\mathbf{v}_i(k+1)$
$\mathbf{v}_i(k)$
$\mathbf{v}_i^{c}(k)$
$\mathbf{x}_i(k)$
$\mathbf{x}_i^{pbest}(k)$
$\mathbf{x}_i(k)$ — current solution
$\mathbf{x}_i^{pbest}(k)$ — previous best solution up to k
$\mathbf{x}_i^{nbest}(k)$ — neighborhood best solution
$\mathbf{x}_i(k+1)$ — new solution

Figure 2.1: Illustration of PSO velocity and particle position update for particle $\mathbf{x}_i$ in a two-dimensional search space.

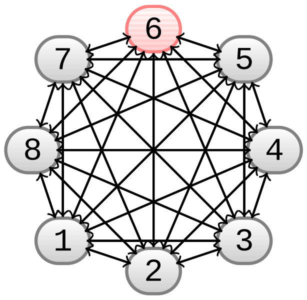
contained in its own neighborhood. Using the adjacency matrix, it is possible to define different types of neighborhood topologies.

In all topologies considered in this work, the locations of the particles in the search space do not affect the neighborhood, as only the particles' indices are required to define the topologies. Here, particle index refers to the position of the particle in the array of particles.

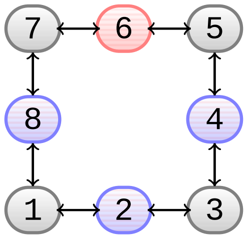
There are several types of PSO neighborhood topologies [2, 3]. The star topology (Fig. 2.2(a)) has only one neighborhood and each particle has a link to every other particle. PSO algorithms using this topology are called 'global best' or 'gbest' algorithms [3, 7, 48, 54]. The use of this topology has been found to lead to rapid convergence, though the algorithm is susceptible to getting trapped in local minima. The adjacency matrix for the star topology is a matrix with ones for all entries.

The topologies shown in Figs. 2.2(b) and 2.2(c) have more than one neighborhood. In the ring topology (Fig. 2.2(b)), each particle has a link to two adjacent particles; thus each neighborhood contains a total of three particles. The neighborhoods in the ring structure are overlapping because each particle resides simultaneously in three neighborhoods. For example, with reference to Fig. 2.2(b), particle 2 is in neighborhoods containing particles 8, 1 and 2; 1, 2 and 3; and 2, 3 and 4. In the cluster topology (Fig. 2.2(c)), the eight particles are placed in two neighborhoods, each containing four particles. PSO algorithms using the ring and cluster neighborhood topologies are called 'local best' or 'lbest' algorithms [3, 7]. In Figs. 2(a), 2(b), and 2(c), the red particle corresponds to the global best and the blue particles represent the local best particles in their respective neighborhoods. The adjacency matrices for the ring and cluster topologies with eight particles are shown in Fig. 2.3.

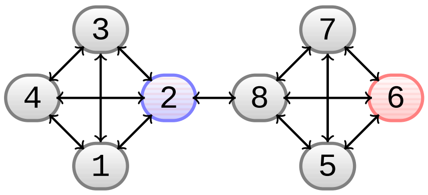
PSO neighborhood topologies can be fixed or they can be varied with iteration. In fixed neighborhood topologies, the links between particles do not change with iteration. The star, ring, and cluster topologies (Fig. 2.2) are examples of fixed topologies. In variable neighborhood topologies, links between particles can vary with iteration. This is achieved by creating new links and by permuting the particle indices [2]. A type of variable neighborhood topology is described next.



(a) Star



(b) Ring



(c) Cluster

Figure 2.2: Examples of PSO neighborhood topologies for a system with eight particles.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

(a) Ring

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

(b) Cluster

Figure 2.3: Adjacency matrices for the ring and cluster topologies shown in Fig. 2.2.

Random variable neighborhood topology

The random variable neighborhood topology is similar to the star neighborhood topology (Fig. 2.2(a)), except that each particle has links to some particles in the swarm but not to other particles. The links between the particles are determined probabilistically. The adjacency matrix for this topology still contains ones on the diagonal; otherwise $m_{ij} = 1$ only when a generated random number is less than a specified probability p, computed as

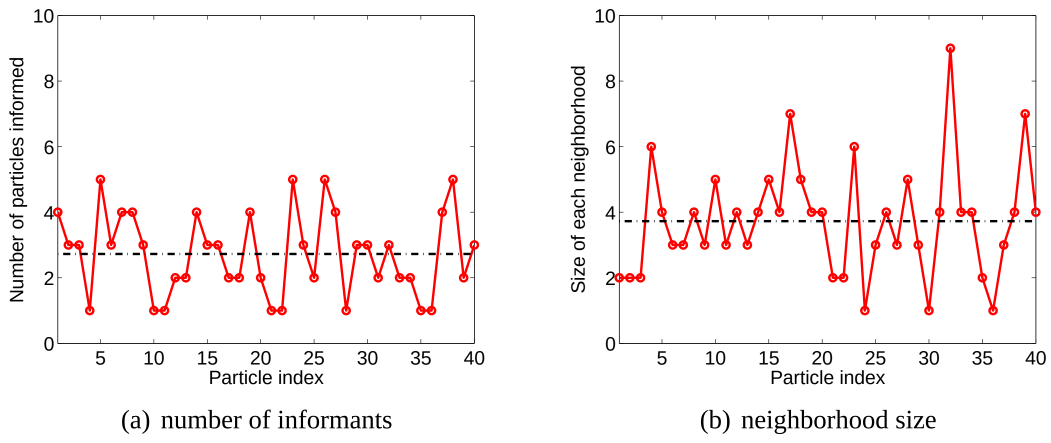


Figure 2.4: Number of particles informed and neighborhood size using Eq. 2.3 with $N_s = 40$ and $N_I = 3$. Note that the number of particles informed and neighborhood size will change when the links are updated.

follows [2]:

$$p = 1 - (1 - 1/N_s)^{N_I}, \tag{2.3}$$

where N_s is the swarm size and N_I is the mean number of particles to be informed. Here p is the probability that a particle is selected randomly (with replacement) after N_I trials to be informed. We take $N_I = 3$, as suggested by [2]. After populating the matrix, the mean number of nonzero elements on any row, i.e., the mean number of neighborhoods to which each particle belongs, is N_I.

The neighborhood topology is updated (by creating new links) if a better solution is not found in the previous iteration. Fig. 2.4 shows an example of the number of particles informed and neighborhood size for each particle using Eq. 2.3 with $N_s = 40$. The dashed line in each plot represents the mean of the plotted quantity. The links and the neighborhood size for each particle change when the links are updated. The random variable topology used here is robust and reduces the susceptibility of solutions to get trapped in local optima.

2.1.2 Treatment of Infeasible Particles

For boundary constrained optimization problems, direct application of Eqs. 2.1 and 2.2 above may cause some particles to leave the feasible region of the search space. To handle these infeasible solutions, we apply the 'absorb' technique [2, 74]. In the absorb technique, invalid particles are moved to the nearest boundary by setting all variables outside the feasible region to their nearest bound (Eq. 2.4). In addition, the affected velocity components are set to zero (Eq. 2.5).

$$x_{i,j}(k+1) = \left\{ \begin{array}{ll} l_j & \text{if } x_{i,j}(k+1) < l_j \\ u_j & \text{if } x_{i,j}(k+1) > u_j \end{array} \right\}, \tag{2.4}$$

$$v_{i,j}(k+1) = 0 \quad \text{if } x_{i,j}(k+1) < l_j \text{ or } x_{i,j}(k+1) > u_j. \tag{2.5}$$

In Eqs. 2.4 and 2.5, l_j and u_j are the lower and upper bounds of the jth component of the search space. Note that Eq. 2.5 is used only after Eq. 2.4 is applied, and the modified velocity is relevant for the computation of $\mathbf{x}_i(k+2)$. Other approaches for handling infeasible particles are discussed in [2, 3, 74, 75], though the strategy described above was found to perform the best in [74, 76].

2.2 Implementation of PSO for Well Placement Optimization

We now describe the specific PSO algorithm used in this work. We used the random neighborhood topology [2, 55] described above. In this case, the best particle among the neighbors of a particle was selected and serves as the local best for that particle. The local best particle was then used in the computation of particle velocity. The absorb technique [74, 76] was employed to handle infeasible particles.

In the applications involving deviated and multilateral wells, the resulting well configurations may be invalid even after applying Eq. 2.4. For example, the main bores of two wells may intersect even though the particle is feasible. To handle such cases, we used a penalty method where we assigned a negative objective function value to particles that result in invalid well configurations. This prevents these particles from being selected as the best particle in their neighborhoods. Other infeasible particle treatment techniques are discussed in [3].

2.2.1 PSO Algorithm Steps

Our implementation of PSO was for parallel computation (all results presented in this section used a cluster of 50 processors). Algorithm 1 presents the steps in the PSO algorithm for a maximization problem. Step 1 initializes the values of ω, c_1, c_2, N_s (swarm size size) and K (number of iterations). The values of ω, c_1 and c_2 are taken from [68]. Step 3 initializes each component of the particle position, $x_{i,j}(k)$, with random elements drawn from a uniform distribution U, $\forall\ j \in \{1, \ldots, D\}$, $\forall\ i \in \{1, \ldots, N_s\}$. Step 4 initializes each component of the velocities, $v_{i,j}(k)$, to zero. Step 5 computes the objective function for all particles. In our application, the objective function was evaluated in all cases by performing a reservoir simulation run (surrogate models were not applied in this work).

Step 6 updates the previous best position for each particle. The particle indices (positions of particles in the array of particles) are permuted in step 9 (this is not required if the random variable topology is used), and the neighborhoods for each particle are generated in step 10. Note that steps 9 and 10 are only performed if the objective function does not improve in the previous iteration. Step 14 determines the best particle in the neighborhood of particle i. New particle velocities, $\mathbf{v}_i(k+1)$, are computed in step 15. Steps 17-21 update all components of the position of particle i. In step 19, infeasible components of the position and the corresponding velocity components are modified using Eqs. 2.4 and 2.5. Step 22 evaluates the objective function $f(\mathbf{x}_i(k+1))$ based on the new particle position. The function evaluations in step 22 are carried out in parallel. Steps 25-32 update the previous best positions for each particle, $\mathbf{x}_i^{pbest}(k)$, if the new objective function value, $f(\mathbf{x}_i(k+1))$, is better than that at the previous best position, $f(\mathbf{x}_i^{pbest}(k))$. The algorithm terminates when a maximum number of iterations is reached.

The algorithm described above is a ‘synchronous’ PSO algorithm [2, 3, 76], in which the previous best particle position for each particle is updated after computing the objective function for all particles. This approach differs from an ‘asynchronous’ PSO algorithm [3], where the previous best particle positions and the best particle in all neighborhoods are updated after computing the objective function of each particle. The asynchronous procedure has better convergence and requires fewer function evaluations [3]. However, the asynchronous approach is a sequential algorithm, while the synchronous approach is better suited for the parallel implementation used here.

2.2.2 Objective Function Evaluation

In all the problems considered, the net present value (NPV) was used as the objective function. Evaluating the NPV of each potential solution requires performing a simulation run. The resulting fluid production profiles generated from the simulation run were used to compute the NPV as follows:

$$\text{NPV} = \sum_{t=1}^{T} \frac{CF_t}{(1+r)^t} - C^{capex}, \tag{2.6}$$

where T is the total production time in years, r is the annual discount rate, C^{capex} is the capital expenditure which represents the total cost to drill and complete all of the wells, and CF_t represents the cash flow at time t. The capital expenditure (C^{capex}) is incurred at time $t = 0$ and is computed as:

$$\begin{aligned} \mathrm{C}^{capex} = \sum_{w=1}^{N^{well}} \Bigg[& C_w^{top} + L_w^{main} C^{drill} \\ & + \sum_{l=1}^{N_w^{lat}} \left[C_l^{junc} + L_{l,w}^{lat} C^{drill} \right] \Bigg], \end{aligned} \tag{2.7}$$

where N^{well} is the number of wells, N_w^{lat} is the number of laterals in well w, C_w^{top} is the cost to drill the main bore to the top of the reservoir (\$), C^{drill} represents the drilling cost within the reservoir (\$/ft), C_l^{junc} is the junction cost of lateral l, L_w^{main} is the length of the main bore (ft), and $L_{l,w}^{lat}$ is the length of lateral l (ft). Eq. 2.7 can be used for any well type, e.g., vertical, deviated, and multilateral wells and can be easily modified if needed to account

Algorithm 1 PSO algorithm

```
1: $\omega = 0.721$, $c_1 = c_2 = 1.193$, define $N_s$, $K$
2: Set iteration index $k = 1$
3: Initialize $\mathbf{x}_i(k)$: $x_{i,j}(k) \sim U(l_j, u_j) \; \forall j, \; \forall i$
4: Initialize $\mathbf{v}_i(k)$: $v_{i,j}(k) = 0 \; \forall j, \; \forall i$
5: Compute objective function, $f(\mathbf{x}_i(k)), \; \forall i$
6: $\mathbf{x}_i^{pbest}(k) = \mathbf{x}_i(k), \; \forall i$
7: **while** $k \leq K$ **do**
8:    **if** No improvement in objective function value **then**
9:       Permute the particle indices
10:      Reinitialize the neighborhoods for each particle
11:   **end if**
12:   $i$ = 1
13:   **while** $i \leq N_s$ **do**
14:      Determine best particle in neighborhood of particle $i$
15:      Compute $\mathbf{v}_i(k+1)$ using Eq. 2.2
16:      $j$ = 1
17:      **while** $j \leq D$ **do**
18:         $x_{i,j}(k+1) = x_{i,j}(k) + v_{i,j}(k+1)$
19:         Apply Eqs. 2.4 and 2.5 if necessary
20:         $j = j + 1$
21:      **end while**
22:      Compute objective function, $f(\mathbf{x}_i(k+1))$
23:      $i = i + 1$
24:   **end while**
25:   **while** $i \leq N_s$ **do**
26:      **if** $f(\mathbf{x}_i(k+1)) > f(\mathbf{x}_i^{pbest}(k))$ **then**
27:         $\mathbf{x}_i^{pbest}(k+1) = \mathbf{x}_i(k+1)$
28:      **else**
29:         $\mathbf{x}_i^{pbest}(k+1) = \mathbf{x}_i^{pbest}(k)$
30:      **end if**
31:      $i = i + 1$
32:   **end while**
33:   $k = k + 1$
34: **end while**
```

for variable C^{drill}.

The cash flow at time t, CF_t, is given by:

$$CF_t = R_t - E_t, \tag{2.8}$$

where R_t and E_t represent the revenue ($) and operating expenses ($) respectively at time t. These quantities depend on the fluid production volumes at time t:

$$R_t = p_o Q_t^o + p_g Q_t^g, \tag{2.9}$$

where p_o and p_g represent the oil price ($/STB) and gas price ($/SCF) and Q_t^o and Q_t^g represent the total volumes of oil (STB) and gas (SCF) produced over the time step. Our models in this chapter do not include gas, in which case $Q_t^g = 0$. The operating expense at time t, E_t, is computed as

$$E_t = p_w^p Q_t^{w,p} + p_w^i Q_t^{w,i}, \tag{2.10}$$

where p_w^p represents water production costs ($/STB), p_w^i represents water injection costs ($/STB), and $Q_t^{w,p}$ and $Q_t^{w,i}$ represent the total volumes of water produced (STB) and injected (STB) respectively at time t. We took p_o, p_g, p_w^p and p_w^i to be constant in time in all cases.

2.3 PSO Applications

We applied the PSO algorithm to several optimization problems. The random variable neighborhood topology was used in these cases. The examples varied in terms of the number and type of wells considered and also in the number of reservoir models included in the optimization (multiple reservoir models were used to incorporate geological uncertainty). Thus the size of the search space and the amount of computation required for function evaluations varied for the different cases. All examples used the NPV as the objective function, with the NPV for each scenario computed using Eq. 2.6. The economic parameters are given in Table 2.1. The simulations were performed using Stanford's General Purpose Research Simulator (GPRS) [77, 78].

In Examples 3 and 4, which involve deviated and multilateral wells respectively, we used the projection well index method [79] to compute the well index for each grid block intersected by the well. This approach accounts for the coupling between the well and the grid block approximately when the well intersects the block at an angle. For representing deviated and multilateral wells within the PSO and GA, we used the parameterization introduced in [28, 20]. Basically, this approach describes wells using the i, j, k location of the well heel, three additional parameters for the toe, a parameter defining the location of junctions in terms of fractional distance along the main bore, and parameters specifying the i, j, k locations of the toes of each branch. Other well representations are also possible, though this representation allowed us to readily constrain the minimum and maximum length of the main bore.

We compared the performance of the PSO algorithm to binary GA, hereafter referred to simply as GA. In [18], different sets of GA parameters were considered and an ‘optimal’ set was identified. We used this optimal set of parameters in our GA runs described in this section. The GA uses the roulette wheel proportional weighting method for selecting individuals, called parents, to generate the new solutions in the next generation. Crossover between two parents generates two new individuals, called offspring. Mutation is then applied to the new individuals. We used single-point crossover and bit-wise mutation in generating new individuals. The GA also employs elitism, where the best individual is carried to the next generation without any modification. Infeasible individuals are assigned negative objective function values (as is also done in PSO), which keeps them from being selected as parents in the next generation. For each of the example problems, we repeated the optimization runs multiple times because of the stochastic nature of both the PSO and GA algorithms. This allowed us to draw more general conclusions regarding the relative performance of the two methods.

We also applied SPSA for several cases and found that it was outperformed consistently by PSO and GA. This observation is in contrast to the findings of [8] and may be due to the need for different SPSA parameters for the types of problems considered here. In a limited study, we attempted to determine appropriate SPSA parameters, but were not able to achieve optimization results using SPSA that were comparable to those using PSO and GA.

Table 2.1: Economic parameters for NPV computation

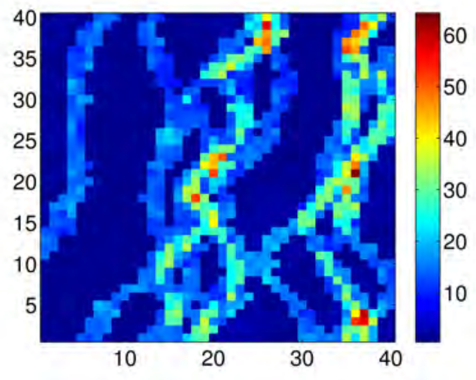
Drilling cost (to reservoir top), C_w^{top}	50×10^6 (\$)
Lateral junction cost, C^{junc}	1.5×10^6 (\$)
Drilling cost per foot, C^{drill}	10,000 (\$/ft)
Oil price, p_o	45 (\$/STB)
Water production cost, p_w^p	10 (\$/STB)
Water injection cost, p_w^i	10 (\$/STB)
Discount rate, r	0.10

2.3.1 Example 1 - Optimizing the placement of a single producer well

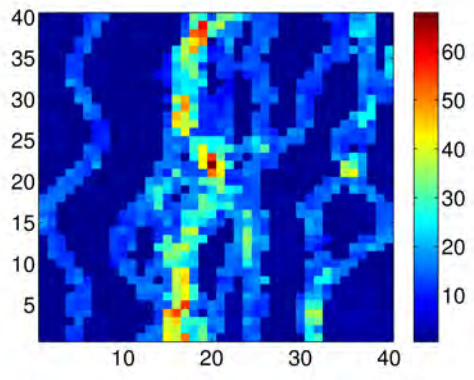
In this example, we determined the optimal location of a single vertical production well. Ten geological realizations were considered, and the objective function was the expected NPV, designated $\langle\text{NPV}\rangle$, computed as $\langle\text{NPV}\rangle = (1/N)\Sigma_{i=1}^{N}\text{NPV}_i$, where NPV_i is the NPV of realization i (with $N = 10$ in this case). The reservoir model is two-dimensional and contains 40 $\times$ 40 grid blocks, with each block of dimensions 300 ft $\times$ 300 ft $\times$ 50 ft. Four realizations of the channelized permeability field are displayed in Fig. 2.5. Porosity was taken to be constant and equal to 0.25. The system contains oil and connate water. As there is no aquifer and no water injection, only oil is produced. Relative permeabilities were not required for this example. The oil viscosity μ_o is 1.20 cp and oil compressibility c_o is 2.0 $\times10^{-5}$ psi^{-1}. Rock compressibility c_r is 3.0 $\times10^{-6}$ psi^{-1}. The system is initially at a pressure of 4800 psi. The production well operates under a BHP constraint of 1000 psi. Total production time is 5000 days.

For this example, there are only 1,600 possible well locations. Thus, by performing 16,000 simulations (one for each well location in each realization), we could sample the search space exhaustively and construct the full objective function surface. This surface, shown in Fig. 2.6, is highly discontinuous and contains many local maxima. This is because of the high level of heterogeneity in these channelized models. The global optimum well location occurs at $i = 20$, $j = 20$. Although we would expect the optimum well location to occur somewhere near the middle of the model, it is fortuitous that this optimum lies right at the center. The corresponding optimal $\langle\text{NPV}\rangle$ is \$268$\times10^6$.

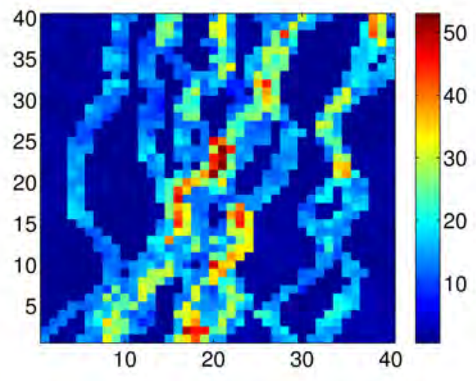
We performed a sensitivity analysis to study the performance of PSO and GA using different swarm (population) sizes and numbers of iterations (generations). Note that, in performing this sensitivity study, we did not need to perform any additional flow simulations, as we had already computed (and saved) the full objective function in constructing Fig. 2.6. We considered swarm/population sizes (N_s) and number of iterations/generations (K) of 5, 10, 20, 30 and 40. We performed 20 optimization runs for each combination of N_s and K. All tests were performed for both PSO and GA procedures.



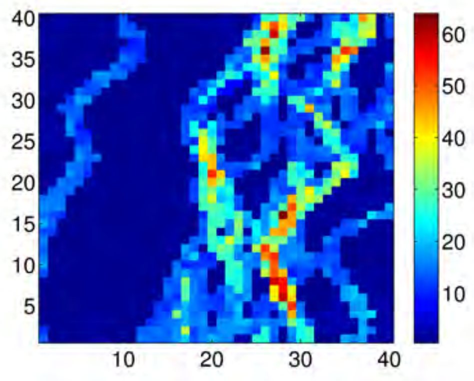
(a) Realization 1



(b) Realization 2



(c) Realization 3



(d) Realization 4

Figure 2.5: Four realizations of the channelized permeability field used in Example 1.

Fig. 2.7 shows a comparison of $\langle \mathrm{NPV} \rangle$ as a function of the total number of simulations per realization ($= N_s \times K$) for both the PSO and GA procedures for N_s of 5, 20 and 40 and K of 5, 20 and 40. The solid red line corresponds to the GA solution, the blue dash-dot line to the PSO solution (both the GA and PSO solutions are averaged over 20 runs), and the

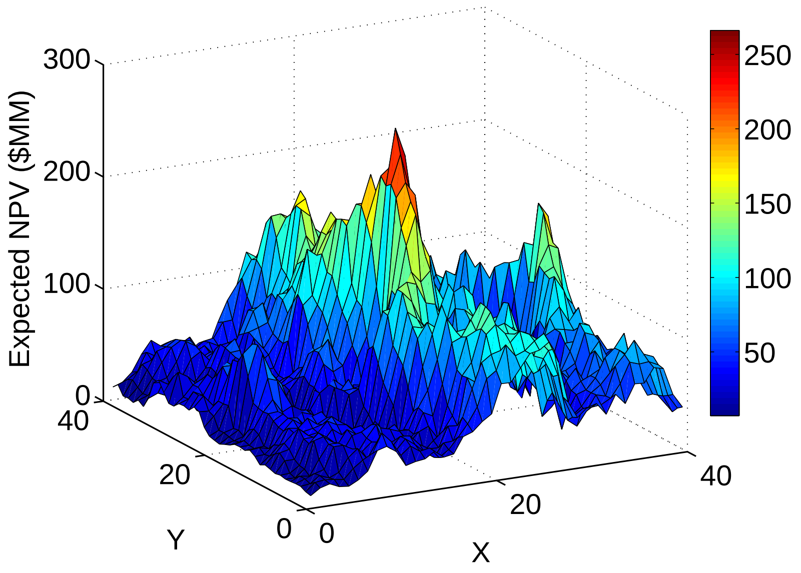


Figure 2.6: Objective function surface for Example 1.

solid line with circles to the global optimum. It is evident that the PSO method performed as well or better than the GA for nearly all combinations of N_s and K considered; i.e., for the same number of function evaluations, PSO provided comparable or better solutions than GA. The advantage of PSO over GA is most noticeable for a swarm (population) size of 5, in which case relatively few total simulations are required. For example, with $N_s = 5$ and $K = 40$ (200 simulations/realization), the PSO solution achieves 94.3% of the global optimum, while GA achieved only 76.4% of this optimum. For large swarm (population) size ($N_s = 40$), the performance of the two methods was essentially the same. Both algorithms converged to the global optimum after approximately 500 simulations per realization.

Note that we used different random seeds in performing the optimizations for each set of N_s and K. Therefore, in Fig. 2.7, for a given row of figures (all of which correspond to the same value of K), the PSO and GA results for small N_s (left column) do not exactly correspond to the early portions of the results for larger N_s (center and right columns).

Figs. 2.8(a) and 2.8(b) show the converged well locations for runs with $N_s = 20$ and

$K = 5$ for the PSO and GA algorithms respectively. Each point corresponds to the converged well location from one of the 20 optimization runs (several of the points coincide or are very close so fewer than 20 distinct points appear). For these parameter values, the PSO solutions are seen to be clustered in the vicinity of the optimal well location ($i = 20$, $j = 20$), in contrast to the GA solutions which show more scatter. For $K = 40$, however, the converged well locations cluster around the optimal well location, with less scatter observed for both the GA and PSO algorithms (Figs. 2.8(c) and 2.8(d)).

2.3.2 Example 2 - Optimizing 20 vertical wells

In this example, we maximized NPV by optimizing the type and location of 20 vertical wells in a two-dimensional reservoir model. Only a single realization of the reservoir was considered in this case. The grid contains 100 $\times$ 100 grid blocks, each of size 300 ft $\times$ 300 ft $\times$ 50 ft. The permeability field is spatially correlated in the southeast-to-northwest direction. The porosity, which is correlated with permeability, varies from block to block. This example involves production and water injection wells. Here, the water viscosity μ_w is 0.31 cp, oil viscosity is 1.20 cp, residual oil and connate water saturations are 0.2, and the relative permeability endpoints (k^0_{ro} and k^0_{rw}) are 0.875 and 0.30 respectively. Corey coefficients for the oil and water curves are approximately 2.5 and 2.9 respectively. Injection wells are specified to operate at 2000 psi, while producers operate at 1000 psi. The other system properties are the same as in Example 1. The total production time is 2000 days.

There are three optimization variables per well, $\{I, \xi, \eta\}$, resulting in a total of 60 variables. The variables ξ and η designate well location and I is an indicator variable that represents the well type ($I = 0$ designates a production well and $I = 1$ an injection well). An indicator variable of this type was used in [19, 28]. The swarm (population) size and maximum number of iterations are 50 and 100 respectively. These values were chosen based on suggestions in [80, 81]. We performed optimization runs for the PSO and GA algorithms.

Fig. 2.9 shows the optimal NPV for each run, as well as the average of these optimal NPVs, versus the number of simulations. The average PSO solution gave an NPV of

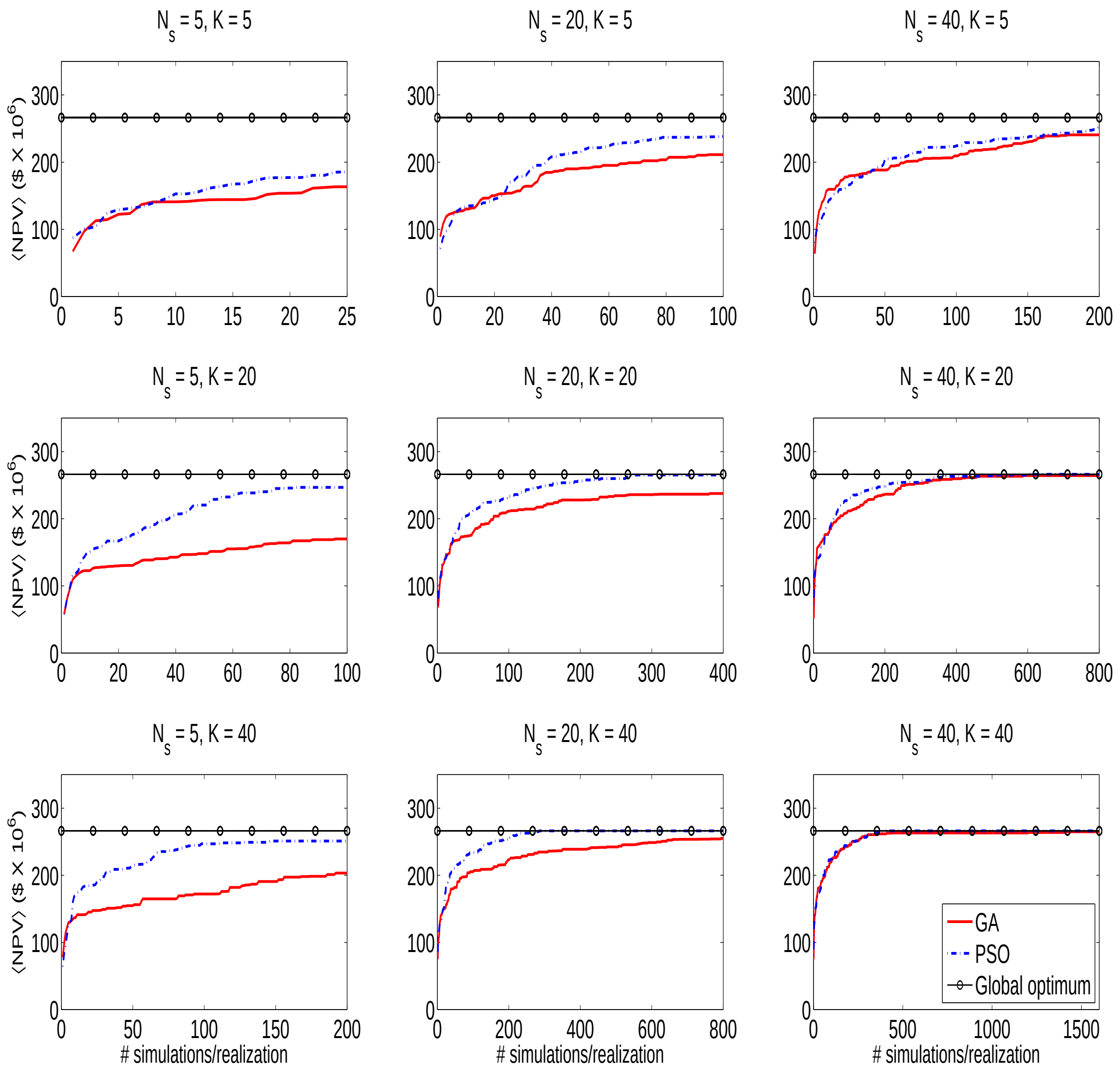
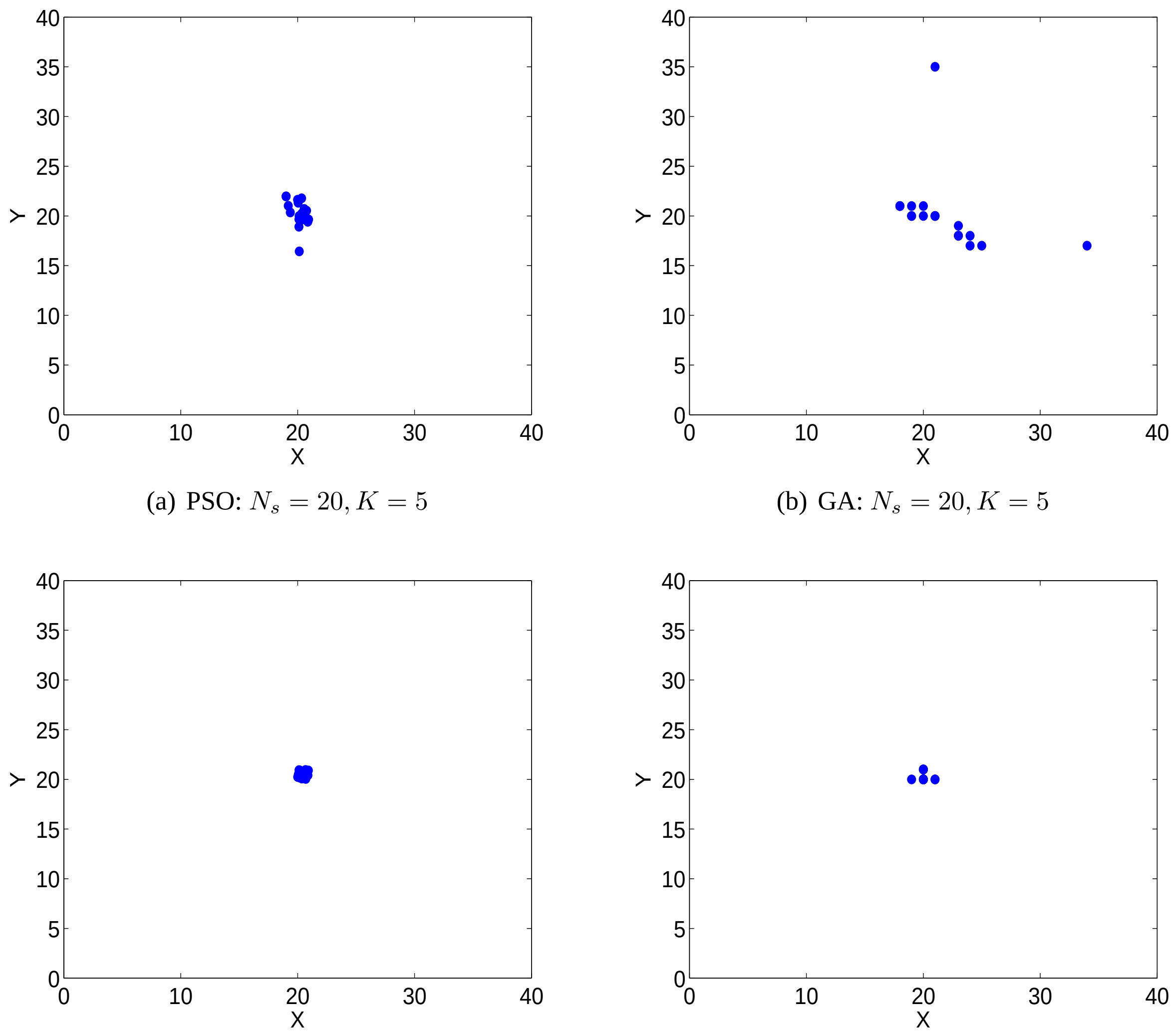


Figure 2.7: ⟨NPV⟩ as a function of number of simulations per realization for PSO and GA for Example 1. ⟨NPV⟩ represents the average over 20 optimization runs.

$\$2.05\times10^9$ compared to an average of $\$1.85\times10^9$ for GA. Although PSO gave an average NPV that is 11% higher than that of the GA, it is evident that one of the PSO runs



(a) PSO: $N_s = 20, K = 5$

(b) GA: $N_s = 20, K = 5$

(c) PSO: $N_s = 20, K = 40$

(d) GA: $N_s = 20, K = 40$

Figure 2.8: Optimal well locations from PSO and GA for $N_s = 20$, and for number of iterations $K = 5$ (top row), and $K = 40$ (bottom row). Each point corresponds to the well location from one of the 20 optimization runs (Example 1).

gave a fairly low NPV. This highlights the need for multiple runs. Note that, in these and subsequent results, during the course of the optimizations, around 5% of the scenarios had invalid well configurations. For these cases we did not perform flow simulations, but we still included them in the total simulation count.

The optimal well locations for the highest NPV case using PSO are shown in Fig. 2.10(a).

The analogous GA result is presented in Fig. 2.10(b). The permeability field is also shown in these plots. It is evident that the wells were not generally located in low permeability regions. Although the exact well locations differed significantly between the two optimizations, it is interesting to note that there were many more producers (open white circles) than injectors (solid black circles) in both cases. In addition, there did not appear to be any sort of well pattern arrangement emerging in either case. Finally, it is evident that some of the well locations were very close. This could be addressed through the use of well-to-well distance constraints.

We compared the optimized NPV to the NPVs obtained for specific well patterns. The NPV for a repeated five-spot arrangement containing a total of 20 wells was \$$1.4\times10^9$, which is considerably less than the optimal NPV achieved by PSO (\$$2.05\times10^9$). Interestingly, however, for an inverted 13-spot arrangement (containing a total of 13 wells), the NPV was slightly higher (\$$2.1\times10^9$) than the optimized NPV for the 20 well case. This suggests that 20 wells are not needed for this problem and that the number of wells should also be an optimization parameter (this issue will be addressed in Chapter 3). The higher NPV for the 13 well case is likely due to the high drilling cost specified in this problem (\$$50\times10^6$ to drill to the top of the reservoir).

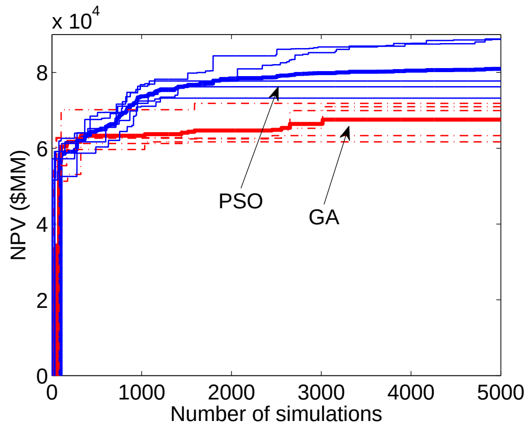


Figure 2.9: NPV of the best solutions versus the number of simulation runs for PSO (blue curves) and GA (red curves) for Example 2. Thin lines correspond to individual runs, while thick lines represent averages over the four runs.

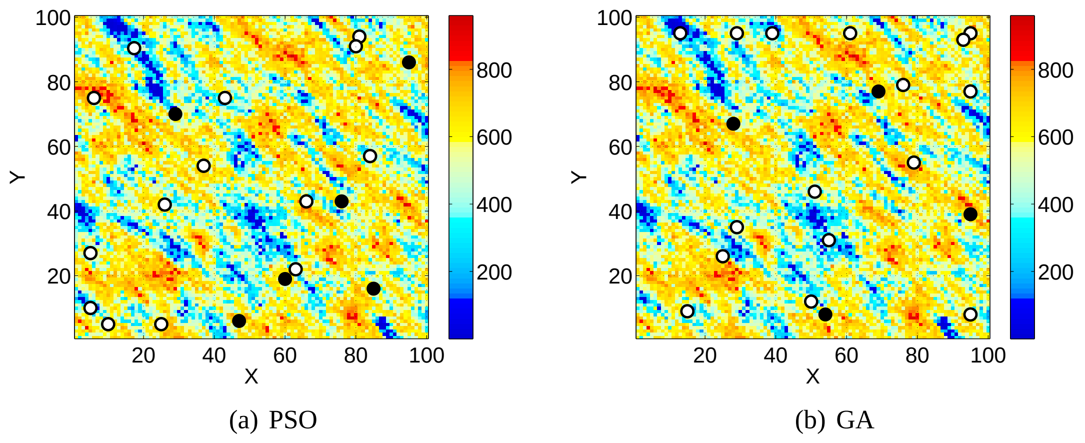


Figure 2.10: Permeability field and well locations from the best runs using PSO and GA for Example 2. The open white circles correspond to producers and the solid black circles to injectors.

2.3.3 Example 3 - Optimizing four deviated producers

This example entails the determination of the optimal location for four deviated monobore producer wells. The reservoir model contains $63 \times 63 \times 7$ grid blocks, each of dimensions 60 ft $\times$ 60 ft $\times$ 20 ft. The other reservoir properties are the same as in previous examples. We consider only a single realization of the reservoir; the permeability field is shown in Fig. 2.11.

Each deviated well is represented by six optimization variables, resulting in a total of 24 variables. The maximum possible length of any well is specified to be 2500 ft. Additional constraints are also included. For example, if the main bores for any two wells intersect, the solution is considered invalid (see Section 2.1.2 for explanation of our treatment of invalid solutions).

We performed five optimization runs for each algorithm. Fig. 2.12 shows the optimal results for each run and the average of these solutions versus the number of simulations. The averaged NPV for the PSO and GA algorithms are \$$479\times10^6$ and \$$448\times10^6$ respectively. Thus the average PSO solution is 7% higher than the average GA solution. The converged well locations from the best run for both algorithms are shown in Fig. 2.13. The results consist of deviated and nearly horizontal wells, though the two solutions differ in terms of the well locations and trajectories.

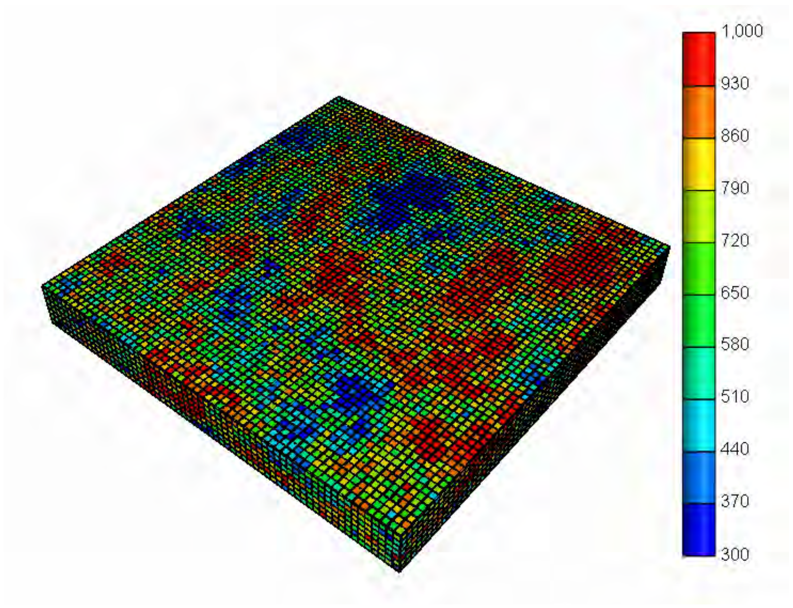


Figure 2.11: Permeability field for Example 3.

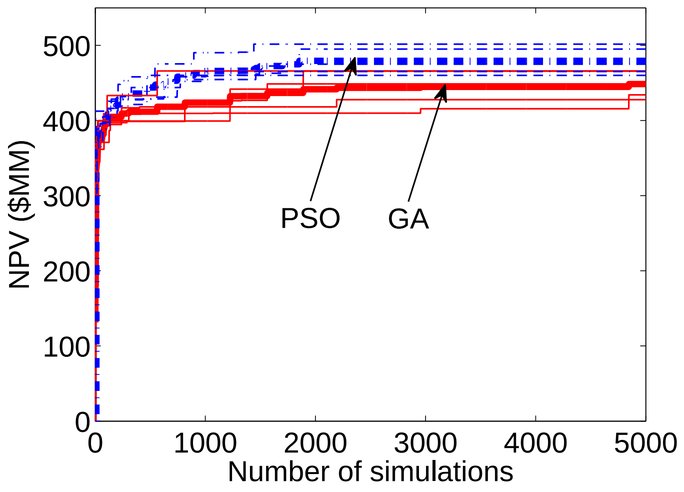
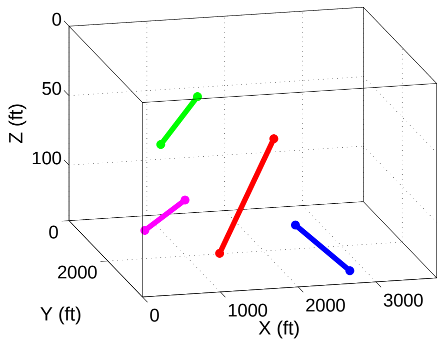
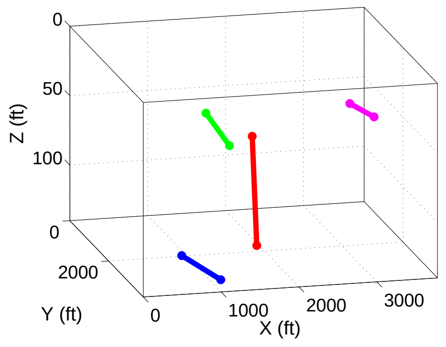


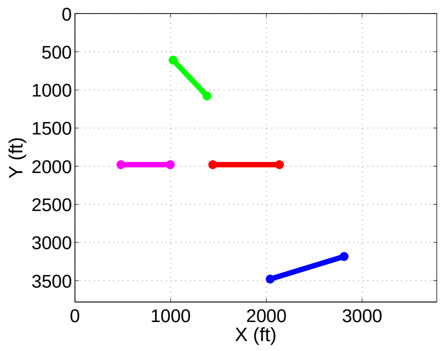
Figure 2.12: NPV of the best solutions versus number of simulation runs for PSO (blue curves) and GA (red curves) for Example 3. Thin lines correspond to individual runs, while thick lines represent averages over the five runs.



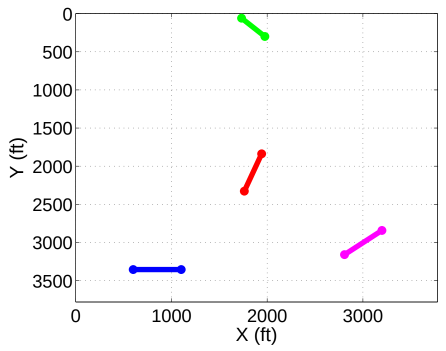
(a) GA - 3D view



(b) PSO - 3D view



(c) GA - top view



(d) PSO - top view

Figure 2.13: Optimum locations for four deviated production wells using GA and PSO (Example 3).

2.3.4 Example 4 - Optimizing two nonconventional producers

In our final example, the type of well was not specified explicitly but was determined as part of the optimization. We optimized the location of two wells and allowed each well to have zero, one or two laterals. The reservoir contains $50 \times 50 \times 10$ grid blocks of dimensions 60 ft $\times$ 60 ft $\times$ 20 ft. The other model properties are the same as in Examples 1, 2 and 3. Fig. 2.14 shows the permeability field for this case (a single reservoir model is considered).

In this problem, the maximum length of the main bore is 2500 ft and the maximum length of any lateral is 1000 ft. The parameterization of each well requires 16 variables, for a total of 32 optimization variables. This problem is more difficult than Example 3 because of additional constraints on the trajectory of the dual laterals. Specifically, well configurations with intersecting main bores and/or laterals are regarded as invalid.

We performed five optimization runs for each algorithm. Fig. 2.15 shows the optimum NPV for each run and the average of these results versus the number of simulations for the PSO and GA algorithms. The average NPV values for the PSO and GA algorithms are $151\times10^6 and $127\times10^6 respectively. The average PSO solution is thus 19% higher than the average GA solution for this case. Fig. 2.16 shows the converged well locations from the best runs for both algorithms. It is interesting to note that both procedures gave solutions containing monobore wells without laterals (though the well locations differ between the two cases). Wells with one and two laterals did exist within the swarm/population, but due to the higher cost of these wells, they did not represent the optimal solutions. For example, in Fig. 2.17, we show an intermediate PSO solution that includes a dual-lateral well. We note finally that in runs with different cost parameters, in which wells were less expensive, optimum solutions did contain dual-lateral wells.

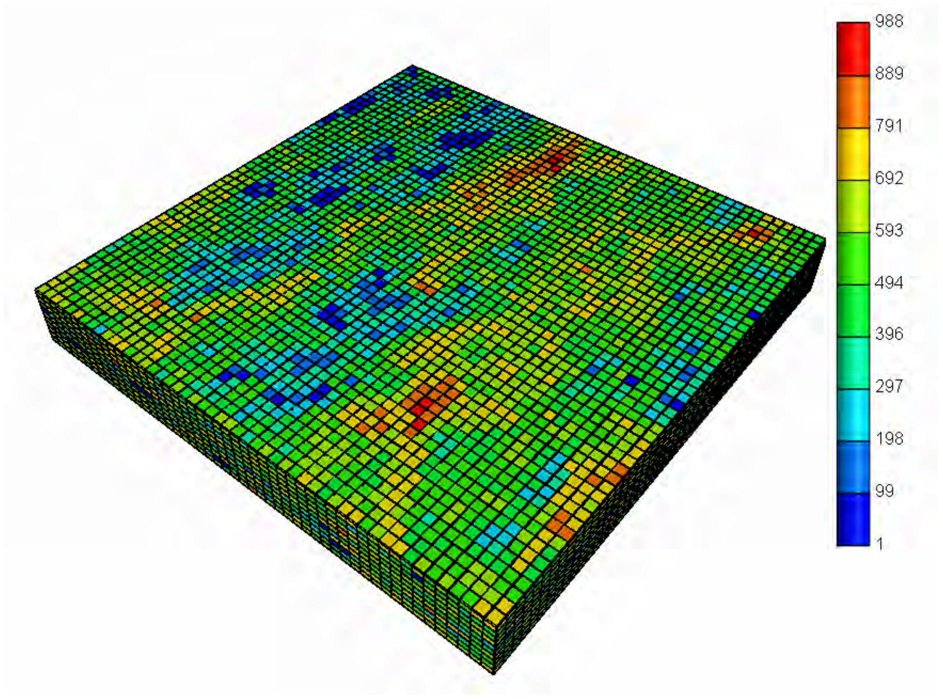


Figure 2.14: Permeability field for Example 4.

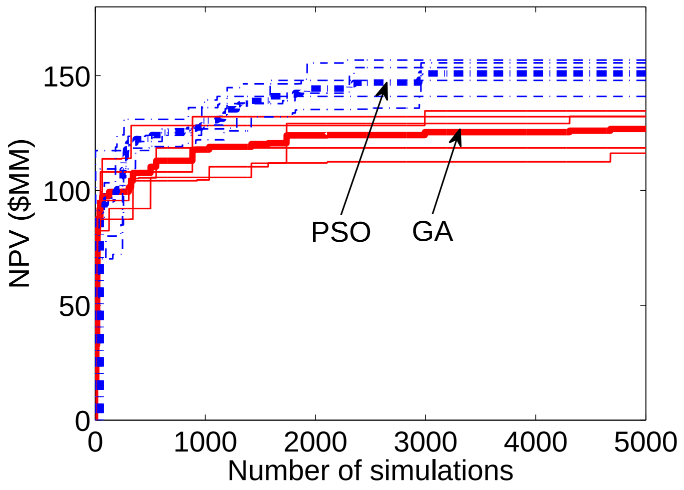
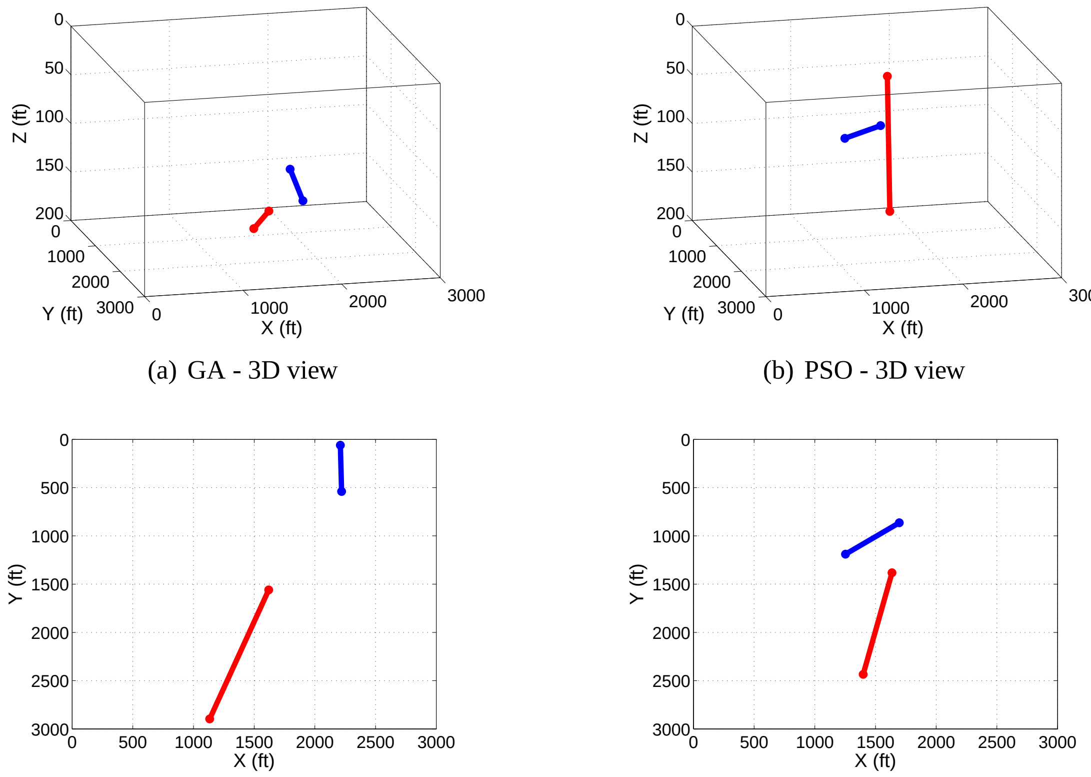


Figure 2.15: NPV of the best solutions as a function of the number of simulation runs for PSO (blue curves) and GA (red curves) for Example 4. Thin lines correspond to individual runs, while thick lines represent averages over the five runs.

2.4 Summary

In this chapter, we described the application of the PSO algorithm to optimize the placement of new wells in oil field development. We considered a variety of optimization problems, involving vertical, deviated, and dual-lateral wells and single or multiple reservoir models, with the maximization of NPV as the objective function. In all the example problems, we demonstrated that the PSO algorithm provided comparable or better results on average than the bGA. However, there are still some issues to be addressed.

In all the example problems, the solutions were represented by a simple concatenation of the well-by-well optimization parameters. There are some problems with this approach. First, the approach can result in invalid well configurations. For example, two vertical wells may occupy the same grid block, or be very close (see Fig. 2.10). Well-pair distance constraints can be incorporated into the optimization procedure to deal with these issues. However, this will increase the difficulty of the optimization problem. Another issue with concatenating the well-by-well parameters is that for problems containing many wells, the number of variables and the size of the search space can become very large.

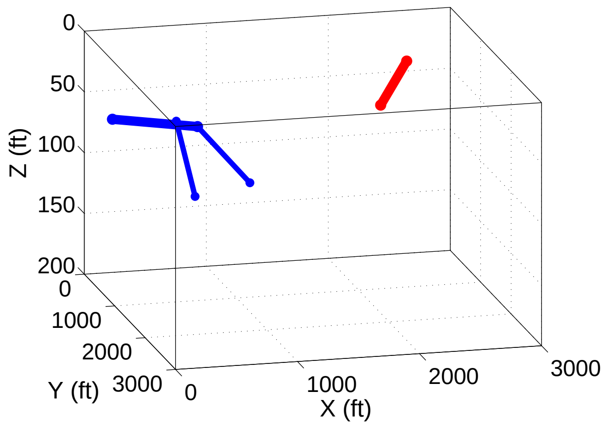


(a) GA - 3D view (b) PSO - 3D view (c) GA - top view (d) PSO - top view

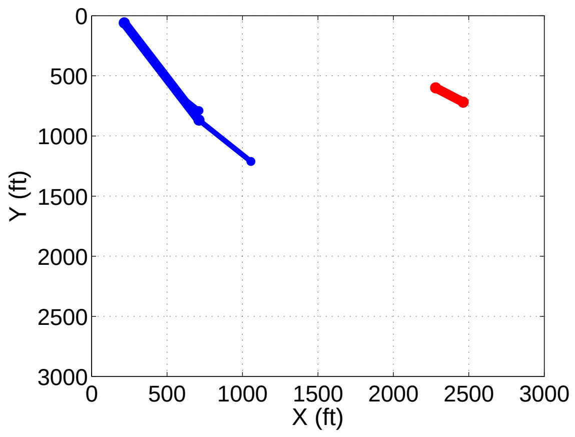
Figure 2.16: Optimum locations for two production wells using GA and PSO (Example 4).

The PSO parameters (ω, c_1, c_2) and neighborhood topology applied here were those suggested in [68, 2] and were not tuned for the particular optimization problems considered. It is likely that the performance of the PSO algorithm could be improved by optimizing the PSO parameters and neighborhood topologies. This issue will be addressed later in Chapter 4.

In the next chapter, we introduce a well pattern optimization algorithm where PSO particles represent full well patterns rather than individual wells. The new procedure enables the optimization of field development problems with a large number of wells. In addition, the new procedure results in practical well configurations and does not require the use of



(a) PSO - 3D view



(b) PSO - top view

Figure 2.17: Intermediate PSO result showing a dual-lateral well and monobore well (Example 4).

well-pair distance constraints.

Chapter 3

Well Pattern Optimization

In this chapter, we present a new procedure for optimizing large-scale field development, called well pattern optimization (WPO). WPO consists of three main components: well pattern description (WPD), well-by-well perturbation (WWP), and the PSO algorithm. In this chapter, we first describe the WPD, which is a novel representation for well patterns. The components of WPD, which include a generic representation for different well pattern types and applicable pattern operators (rotation, scale, shear and switch), are discussed. Next, we present the well-by-well perturbation (WWP) which can be used to improve the solutions obtained using WPD. Several example problems were considered using the WPO procedure. In total, these results demonstrate the applicability of WPO for large-scale field development.

3.1 Well Pattern Description (WPD)

The overall well pattern optimization (WPO) algorithm contains as key components the well pattern description (WPD), well-by-well perturbation, and the core optimization algorithm (PSO in this case). The WPD, which we now describe, treats well patterns (rather than individual wells) as the basic unit of representation. Thus, in our PSO implementation, each particle represents a repeated well pattern. WPD can encode representations for a wide variety of possible well patterns, in addition to the transformations that are used to manipulate these well patterns. It is the parameters that define the patterns and quantify the transformations that are optimized during the course of the optimization. The WPD representation offers several benefits including a reduction in the number of optimization variables, the ability to perform optimizations without well-to-well distance constraints, and the automatic determination of the optimum number of wells. The use of well patterns also presents some challenges. For a robust optimization, many different well pattern types, shapes and orientations must be considered, and there is a very large number of possible combinations of these attributes. Thus a concise solution representation, coupled with an efficient and robust core optimization algorithm, is required for this problem.

In the WPD representation, each solution contains three groups of optimization parameters: the basic parameters associated with each of the different well patterns, parameters that quantify the pattern operations, and parameters that define the sequence of application of these operations. In the following sections we will describe these three elements in detail.

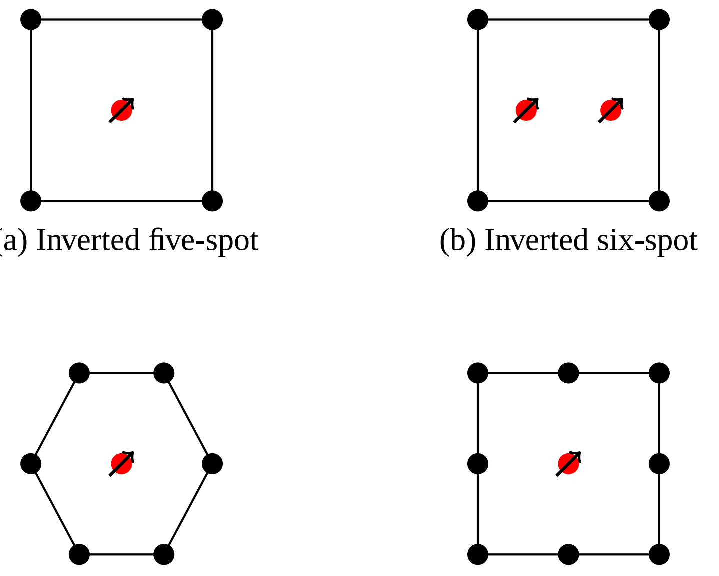
3.1.1 Basic Well Pattern Parameters

In order to consider different well pattern types in the optimization, a basic well pattern representation is required. For this purpose, we extend the representation for inverted five-spot patterns described in [39]. Our representation uses four variables to represent the well pattern: $\{\xi^0, \eta^0, a, b\}$, where (ξ^0, η^0) designate the areal location of the center of the pattern (a well may or may not be located at (ξ^0, η^0)) and a and b specify well spacings. We represent areal location using (ξ, η) rather than the usual (x, y) because $\mathbf{x}$ is used to designate PSO solutions. Our extended representation is as follows:

$$\mathcal{P} = \{I^{wp}, [\xi^0, \eta^0, a, b]\}, \tag{3.1}$$

where I^{wp} is an integer variable that defines the pattern type (e.g., seven-spot, nine-spot). If we consider N_p different well pattern types in the optimization, then $I^{wp} \in \{1, 2, \ldots, N_p\}$, with each value corresponding to a different pattern type. The representation shown in Eq. 3.1 is quite simple and is applicable for many pattern types. This representation could be readily extended to account for more complex well arrangements such as 13-spot patterns.

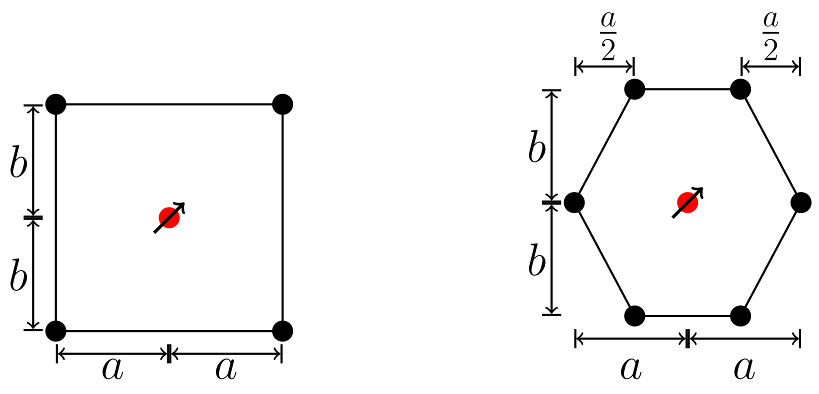
Several well patterns, containing different numbers of wells, are shown in Fig. 3.1. It is evident that, using the representation shown in Eq. 3.1, each pattern can be represented in terms of the five variables appearing in the equation (see Fig. 3.2). Were we to represent each well individually, a five-well pattern would require ten parameters (ξ and η locations of each well) and a nine-well pattern would require 18 parameters. Thus the representation in Eq. 3.1 clearly leads to significant reduction in the dimension of the search space.


(a) Inverted five-spot (b) Inverted six-spot

(c) Inverted seven-spot (d) Inverted nine-spot

Figure 3.1: Illustration of different types of well patterns. The solid black circles represent producers and the red circles with arrows represent injectors.

In our algorithm, well patterns are in all cases repeated to fill the entire reservoir domain. Each pattern has the same size and orientation as the base pattern. Wells that fall outside of the reservoir boundaries are eliminated from the total set. The well pattern parameters a and b, in addition to the parameters connected to the operators, thus determine the total number of wells associated with each PSO solution (particle). In this way, WPO determines the optimal number of wells. We constrain the minimum and maximum values of a and b such that the patterns they define are of physically reasonable sizes. The bounds prescribed for a and b depend on the bounds used for the parameters associated with the pattern operators, as these also impact the size of the patterns.



(a) Inverted five-spot (b) Inverted seven-spot

Figure 3.2: Well pattern element representations for the inverted five-spot and seven-spot patterns.

The well pattern representation in Eq. 3.1 does not allow for the general orientation of well patterns. In fields with large-scale permeability features or correlations, this representation may be suboptimal since patterns cannot align themselves to take advantage of trends in flow. We now describe techniques that generalize the representation given in Eq. 3.1.

3.1.2 Well Pattern Operators

Well pattern operators define operations that can be performed on the encoded well patterns. When applied to a pattern, these operators can alter the pattern size, shape, orientation, type (normal versus inverted) and the location of the wells in the pattern. We developed four well pattern operators: rotation, scale, shear, and switch operators. The rotation operator rotates a well pattern, the scale operator increases or decreases the size of a well pattern, the shear operator skews the shape of a well pattern, and the switch operator changes the pattern type from the normal to inverted form by switching production wells to injection wells and vice versa. Other operators can be readily incorporated into the WPD representation. In general, application of these pattern operators requires the specification of several parameters including the reference well. The reference well serves as the origin for the pattern operation and its location remains unchanged after the operation is performed.

We now define the pattern operators and associated parameters. Each pattern has wells located at the vertices of a polygon (outer wells) and, except for the six-spot pattern, there is also a well at the center (Fig. 3.1). In our numbering convention, the outer wells are numbered consecutively in the anticlockwise direction followed by the interior well(s).

Each pattern operator takes as input a well pattern and produces a new well pattern as output. We designate $\mathbf{W}_{in}$ to be an $N \times 2$ matrix representing the well locations in the input well pattern element (these well locations are designated (ξ, η)) and $\mathbf{W}_{out}$ to be the corresponding matrix representing the output well locations (designated $(\hat{\xi}, \hat{\eta})$). In both matrices, well locations are relative to the reference well, located at (ξ^{ref}, η^{ref}). The two matrices are given by:

$$\mathbf{W}_{in} = \begin{pmatrix} \xi_1 - \xi^{ref} & \eta_1 - \eta^{ref} \\ \xi_2 - \xi^{ref} & \eta_2 - \eta^{ref} \\ \vdots & \vdots \\ \xi_n - \xi^{ref} & \eta_n - \eta^{ref} \\ \vdots & \vdots \\ \xi_{N_{wp}} - \xi^{ref} & \eta_{N_{wp}} - \eta^{ref} \end{pmatrix}, \; \mathbf{W}_{out} = \begin{pmatrix} \hat{\xi}_1 - \xi^{ref} & \hat{\eta}_1 - \eta^{ref} \\ \hat{\xi}_2 - \xi^{ref} & \hat{\eta}_2 - \eta^{ref} \\ \vdots & \vdots \\ \hat{\xi}_n - \xi^{ref} & \hat{\eta}_n - \eta^{ref} \\ \vdots & \vdots \\ \hat{\xi}_{N_{wp}} - \xi^{ref} & \hat{\eta}_{N_{wp}} - \eta^{ref} \end{pmatrix}, \tag{3.2}$$

where N_{wp} is the number of wells in the well pattern. Most of the well pattern transformations can now be described through the following operation:

$$\mathbf{W}_{out}^T = \mathbf{M}\,\mathbf{W}_{in}^T, \tag{3.3}$$

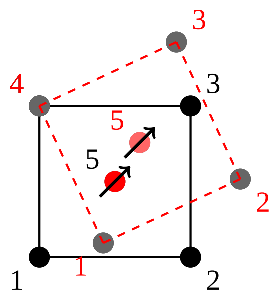
where $\mathbf{M}$ is a 2×2 transformation matrix. The specific forms of $\mathbf{M}$ for the relevant pattern operators are described below. We illustrate the well pattern operators using the inverted five-spot and seven-spot well patterns, though the operators also apply to the other well pattern types.

Rotation operator

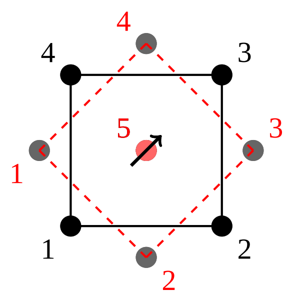
The rotation operator, designated $\mathcal{O}_{rot}$, rotates a well pattern by an angle θ about a reference well, n^{ref}, $n^{ref} \in \{1, 2, \ldots, N_{wp}\}$. After the rotation, the locations of all wells other than n^{ref} are altered. The rotation operator does not change the size of the well pattern. The rotation of the pattern element is achieved through use of $\mathbf{M} = \mathbf{M}_\theta$ in Eq. 3.3, where

$$\mathbf{M}_\theta = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}. \tag{3.4}$$

This results in clockwise rotation for $\theta > 0$ and anticlockwise rotation for $\theta < 0$. Figure 3.3 illustrates the rotation operator applied to an inverted five-spot pattern. In Fig. 3.3(a), the initial well pattern (solid lines) is rotated about well 4 with $\theta = -25°$ (anticlockwise rotation). In the final pattern (dashed lines), the locations of wells 1, 2, 3 and 5 differ from those in the initial pattern. Fig. 3.3(b) shows a $45°$ (clockwise rotation) about well 5.



(a) Rotation about well 4 with $\theta = -25°$



(b) Rotation about well 5 with $\theta = 45°$

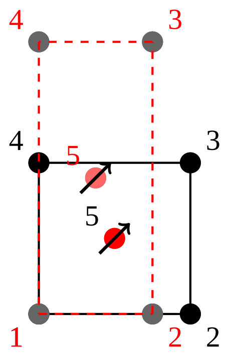
Figure 3.3: Illustration of the rotation operator applied to an inverted five-spot pattern.

Scale operator

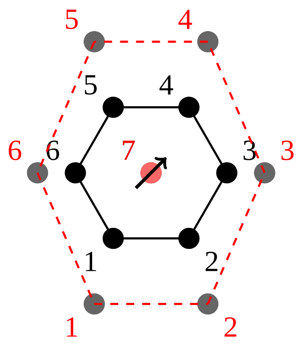
The scale operator, $\mathcal{O}_{scale}$, increases or decreases the size of a well pattern. The scale operator requires as arguments the reference well in the pattern and axis scaling factors for the ξ and η directions. If the scale factor for an axis is greater than 1, the pattern is stretched in that direction. If the scale factor is less than 1, the well pattern is shrunk along that direction. A nonuniform scale matrix, $\mathbf{M}_{sc}$, is used to achieve the scaling of a well pattern:

$$\mathbf{M}_{sc} = \begin{pmatrix} S_\xi & 0 \\ 0 & S_\eta \end{pmatrix}, \tag{3.5}$$

where S_ξ and S_η are axis scaling factors. Figures 3.4(a) and (b) illustrate the scale pattern operator applied to the inverted five-spot and inverted seven-spot patterns. In Fig. 3.4(a) the well pattern is scaled relative to well 1 using scaling factors $\{0.75,\ 1.8\}$. In Fig. 3.4(b), the inverted seven-spot pattern is scaled with factors $\{1.5,\ 2.0\}$ relative to well 7. Because the pattern is replicated over the entire field, it is clear that these scaling parameters will have a strong impact on the total number of wells. In the examples, the scaling factors are constrained to be between 0.5 and 2.



(a) Scaling with factors $\{0.75, 1.8\}$ from well 1



(b) Scaling with factors $\{1.5, 2.0\}$ from well 7

Figure 3.4: Illustration of the scale pattern operator for the inverted five and seven-spot patterns.

Shear operator

The shear operator, $\mathcal{O}_{shear}$, alters the shape of a well pattern by shearing (skewing) the well pattern in the ξ and η directions. The shear operator requires three arguments: a reference well and axis shearing factors for the ξ and η directions. These factors indicate the amount of shearing in each direction relative to the other direction. The shearing of the pattern element is achieved using a shear matrix, $\mathbf{M}_{sh}$:

$$\mathbf{M}_{sh} = \begin{pmatrix} 1 & H_\xi \\ H_\eta & 1 \end{pmatrix}, \tag{3.6}$$

where H_ξ and H_η are axis shearing factors. Care must be taken in defining the minimum and maximum values of H_ξ and H_η, as well locations become colinear if H_ξ and H_η approach -1 or 1. In the examples, the shearing factors are constrained to be between -0.5 and 0.5. Figure 3.5 illustrates the shear operator applied to an inverted five-spot pattern.

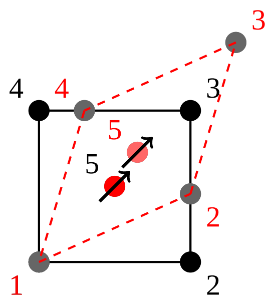
Switch operator

The switch operator, $\mathcal{O}_{switch}$, switches a well pattern from the normal to the inverted form, and vice versa. This is achieved by switching the type (producer, injector) of all the wells in the pattern. The switch operator does not require any arguments.

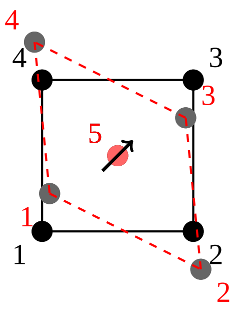
The switch operator offers some benefits for the overall WPO algorithm. The switch operator enables the algorithm to consider both normal and inverted forms of the patterns without increasing the number of patterns that need to be defined in the algorithm. The switch operator also allows the algorithm to consider different producer-injector well ratios for the same well pattern parameters. The producer-injector ratios for the normal and inverted forms of the five-spot, seven-spot, and nine-spot patterns are shown in Table 3.1 [82]. Figures 3.6(a) and (b) illustrate the switch pattern operator applied to the inverted five-spot and inverted seven-spot patterns.

Representation of well pattern operators

As described above, each pattern operator (except the switch operator) requires the specification of a reference well in the pattern and at most two additional operator arguments.



(a) Shearing with factors $\{0.45, 0.3\}$ from well 1



(b) Shearing with factors $\{-0.50, -0.10\}$ from well 5

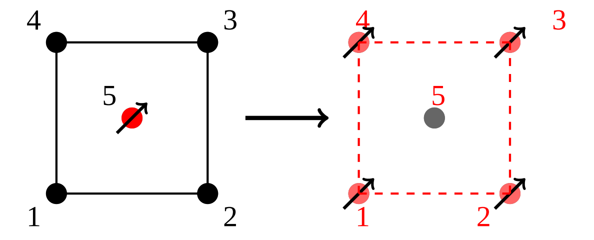
Figure 3.5: Illustration of the shear pattern operator for the inverted five-spot pattern.

Table 3.1: Producer-injector ratios for different well patterns

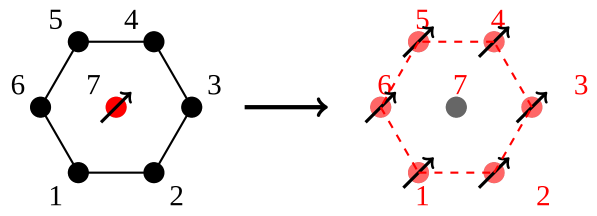
Pattern	Pattern form	
	Normal	Inverted
Five-spot	1	1
Seven-spot	1/2	2
Nine-spot	1/3	3

This allows us to use a generic representation for these pattern operators which can be readily extended to other operators that may be introduced.

Let $\mathcal{O}_j$ represent the jth pattern operator, where $\mathcal{O}_j \in \{\mathcal{O}_{rot},\ \mathcal{O}_{scale},\ \mathcal{O}_{shear},\ \mathcal{O}_{switch}\}$.



(a) Switching an inverted five-spot pattern



(b) Switching an inverted seven-spot pattern

Figure 3.6: Illustration of the switch pattern operator for the inverted five and seven-spot patterns.

In the WPD representation, $\mathcal{O}_j$ is represented as:

$$\mathcal{O}_j = [\ \underbrace{\{n_j^{ref}\}}_{\text{reference well}}\ , \underbrace{\{\arg_{j,1}, \arg_{j,2}\}}_{\text{operator arguments}}] \tag{3.7}$$

where n_j^{ref} is the reference well for operator j and $\{\arg_{j,1}, \arg_{j,2}\}$ is the list of arguments for the operator.

In our implementation, the arguments appearing in Eq. 3.7 are represented as normalized variables between 0 and 1. Each variable is then rescaled as required before being used in the pattern operation. This enables appropriate exchange of information between particles that are acted on by different operators during the course of the particle swarm optimization.

3.1.3 Solution Representation in WPD

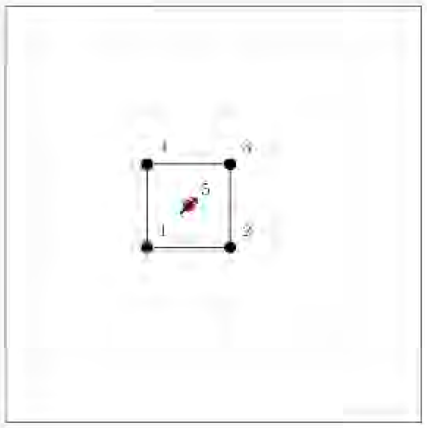
We now describe the overall representation of potential solutions using the well pattern description. Each solution (particle within our PSO implementation) consists of the basic well pattern definition and parameters (Eq. 3.1) and the representation for the pattern operators (Eq. 3.7). In addition, each solution contains a set of variables that defines the sequence of pattern operations when multiple operators are applied. The ith PSO particle, $\mathbf{x}_i$, is thus encoded as:

$$\mathbf{x}_i = [\underbrace{\{I_i^{wp}, [\xi_i^0, \eta_i^0, a_i, b_i]\}}_{\text{pattern parameters}} \underbrace{\{S_{i,1}, S_{i,2}, \ldots, S_{i,\mathcal{N}_o}\}}_{\text{operator sequence}} \underbrace{\{\mathcal{O}_{i,1}, \mathcal{O}_{i,2}, \ldots, \mathcal{O}_{i,\mathcal{N}_o}\}}_{\text{pattern operators}}], \tag{3.8}$$

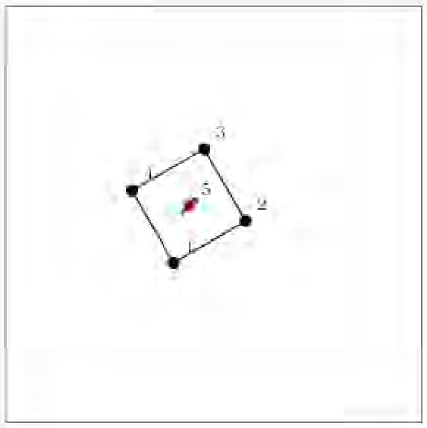
where $\{I_i^{wp}, [\xi_i^0, \eta_i^0, a_i, b_i]\}$ are the pattern parameters for particle i, $\mathcal{N}_o$ is the number of pattern operators, $\{\mathcal{O}_{i,1}, \mathcal{O}_{i,2}, \ldots, \mathcal{O}_{i,\mathcal{N}_o}\}$ provide the list of pattern operators, and $\{S_{i,1}, S_{i,2}, \ldots, S_{i,\mathcal{N}_o}\}$ represent the sequence of application of the pattern operators. Each $S_{i,j}$, $S_{i,j} \in \{0, 1, 2, \ldots, \mathcal{N}_o\}$, is an integer variable representing the index of a pattern operator. For example, if $S_{i,1} = 1$ and $S_{i,2} = 2$, then the pattern operator $\mathcal{O}_{i,1}$ is applied first and pattern operator $\mathcal{O}_{i,2}$ is applied second (using the well pattern generated from $\mathcal{O}_{i,1}$). If $S_{i,j} = 0$, then the jth pattern operator ($\mathcal{O}_{i,j}$) is skipped.

All components of any particle $\mathbf{x}_i$, which represents a PSO solution, are treated as real numbers. However, some of the optimization parameters, such as n_j^{ref} and $S_{i,j}$, are integers. Where necessary, we determine integer values from real values by simply rounding to the nearest integer.

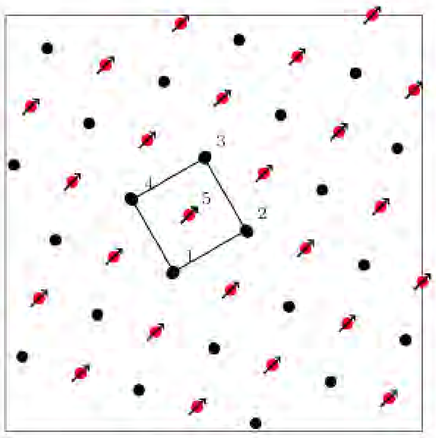
Examples of using one pattern operator and two pattern operators in sequence are illustrated in Figs. 3.7 and 3.8. In Fig. 3.7, the rotation operator is applied to the initial well pattern (Fig. 3.7(a)) and the resulting pattern (Fig. 3.7(b)) is repeated over the entire domain (Fig. 3.7(c)). In the example in Fig. 3.8, the shear and scale operators are applied in sequence to an inverted seven-spot pattern. As indicated above, wells that fall outside the reservoir boundaries are eliminated. It is evident from these figures that a wide variety of well patterns can be generated (and thus evaluated in the WPO procedure) using the WPD representation.



(a) Initial pattern



(b) After rotation

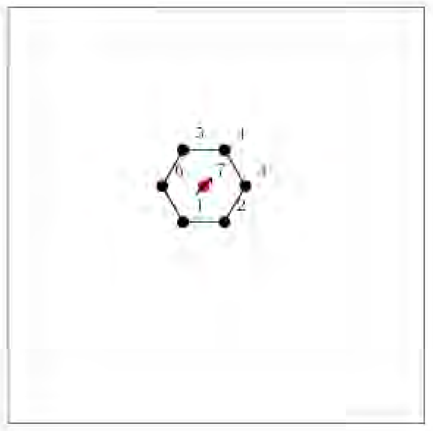


(c) Repeated final pattern

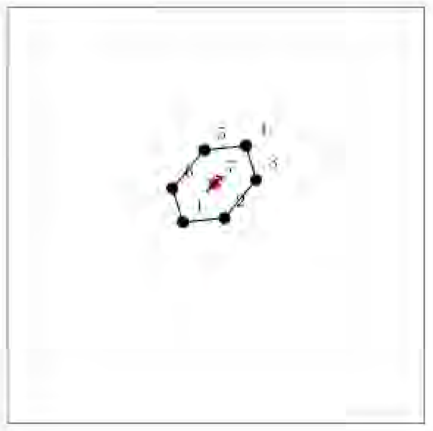
Figure 3.7: Application of one pattern operator.

3.2 Well-by-Well Perturbation (WWP)

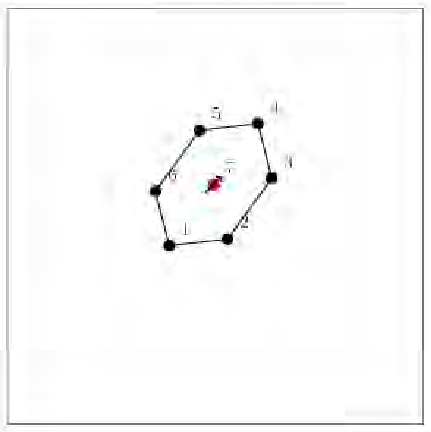
Optimization using WPD produces solutions that consist of repeated well patterns. It is possible to further improve the solution by performing optimizations that involve perturbing the well locations determined using WPD. We refer to this as well-by-well perturbation (WWP). Following WWP, the basic patterns remain essentially intact, but the well locations within the patterns are shifted to improve the objective function.



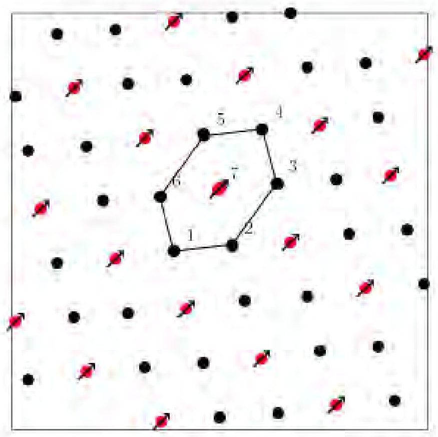
(a) Initial pattern



(b) After shearing



(c) After scaling



(d) Repeated final pattern

Figure 3.8: Application of two pattern operators (shear and scale).

We again use the PSO algorithm for the WWP optimization. Here, however, each PSO particle $\mathbf{x}_i$ contains a concatenation of the perturbations to be applied to the well locations determined from WPD:

$$\mathbf{x}_i = \{\underbrace{\Delta\xi_1,\ \Delta\eta_1}_{\text{well 1}},\ \underbrace{\Delta\xi_2,\ \Delta\eta_2}_{\text{well 2}},\ \ldots,\ \underbrace{\Delta\xi_n,\ \Delta\eta_n}_{\text{well } n},\ \ldots,\ \underbrace{\Delta\xi_N,\ \Delta\eta_N}_{\text{well } N}\}, \tag{3.9}$$

where N is the total number of wells as determined in the WPD optimization and $\Delta\xi_n$ and $\Delta\eta_n$ represent the perturbations of the spatial locations of well n. Prior to simulation, the actual well locations are obtained by adding the perturbations to the corresponding well location from the WPD solution. In our implementation, the minimum and maximum values of $\Delta\xi_n$ and $\Delta\eta_n$ are constrained to keep wells from shifting from one pattern to another. We note that WWP could also be used for other determinations, such as the completion interval for each well in three-dimensional problems. Elimination of wells could also be considered through inclusion of an active/inactive optimization parameter for each well. Neither of these options was considered in this work, though they could be incorporated easily into the WWP procedure (the dimension of the search space will, however, increase).

The WWP procedure introduces an efficient 'local' search, which leads to improved solutions (as will be demonstrated below). Improved solutions are achieved because the optimized well locations now account for local variations in porosity, permeability, and other properties. The two procedures – WPD and WWP – are complementary because WPD provides the ability to search efficiently on the large scale and to optimize well count, while WWP enables local adjustments.

Although the dimension of the search space in WWP is the same as that using well-by-well concatenation (for cases where the number of wells is specified), WWP has several advantages over well-by-well concatenation. Specifically, in WWP, wells are allowed to move only a limited number of grid blocks in each direction. This 'constraint' can be easily incorporated into the optimization, in contrast to the general well distance constraints required using well-by-well concatenation. In addition, because wells can move only locally in WWP, the size of the search space is much smaller than that for well-by-well concatenation (despite the fact that the dimension of the search space is the same in both cases). Finally, in WWP the number of wells (as determined from the WPD optimization) is fixed and does not need to be determined as part of the optimization. Using well-by-well concatenation, however, the number of wells should also be an optimization variable, which further complicates the optimization.

It is important to note that the use of WPD followed by WWP cannot be expected to provide the overall global optimum that could (theoretically) be achieved through use of well-by-well concatenation. This is because well-by-well concatenation entails a broader search, which should ultimately lead to a better global optimum than that resulting from the use of WPD plus WWP. However, the WPD plus WWP search is much more efficient, so this approach is expected to provide better solutions given practical computing resources.

3.3 Examples Using WPO

We now describe the application of the WPO procedure to four example problems. In all cases, we maximized net present value (NPV) using the procedures described above. The economic parameters used for the computation of NPV are provided in Table 3.2. Simulation runs for Examples 1, 3 and 4 were performed using Stanford's General Purpose Research Simulator, GPRS [77, 78]. The 3DSL streamline simulator [83] was used for Example 2. Because of the stochastic nature of the PSO algorithm, we performed multiple optimization runs with the same set of input parameters. This enabled us to gauge the degree of variability in the optimization results. In the figures in this section, 'x-grid' and 'y-grid' refer to grid block locations in the simulation models.

Table 3.2: Economic parameters for NPV computation

Well cost	3×10^6 ($)
Oil price (Examples 1, 3)	60 ($/STB)
Oil price (Examples 2, 4)	80 ($/STB)
Gas price (Example 2)	2.5 ($/MSCF)
Water production cost (Examples 1, 3, 4)	5 ($/STB)
Water injection cost (Examples 1, 3, 4)	5 ($/STB)
Water production cost (Example 2)	10 ($/STB)
Water injection cost (Example 2)	10 ($/STB)
Discount rate, r	0.10

3.3.1 Example 1: WPD optimizations using different numbers of operators

In this example, we performed optimizations using either one or four WPD operators per particle. We optimize using only WPD – the second phase WWP optimization is not applied for this case.

We considered a synthetic, heterogeneous, two-dimensional reservoir model containing 100 $\times$ 100 grid blocks, with each block of dimensions 100 ft $\times$ 100 ft $\times$ 40 ft. The permeability field, shown in Fig. 3.9, was generated geostatistically using an exponential variogram model with oriented correlation lengths of 1000 ft and 5000 ft. Porosity varies from block to block and is correlated with permeability. The reservoir model initially contains oil and water ($S_{oi} = 0.80$, $S_{wi} = 0.20$). The oil viscosity μ_o is 1.20 cp and oil compressibility c_o is 2.0 $\times 10^{-5}$ psi^{-1}. For water we specify μ_w = 0.31 cp and c_w = 2.9 $\times 10^{-6}$ psi^{-1}. Relative permeability end points for oil and water are 0.85 and 0.30 respectively. The initial reservoir pressure is 5000 psi. The production and injection wells operate under BHP constraints of 1000 psi and 6000 psi respectively. The total production time is 3650 days.

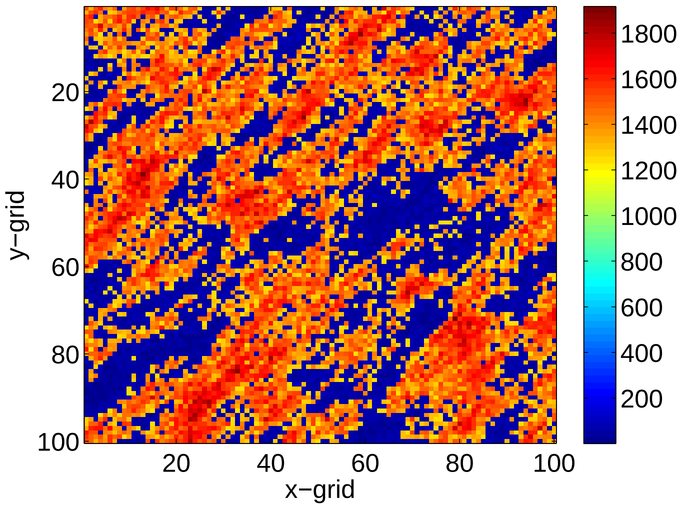


Figure 3.9: Permeability field (Example 1).

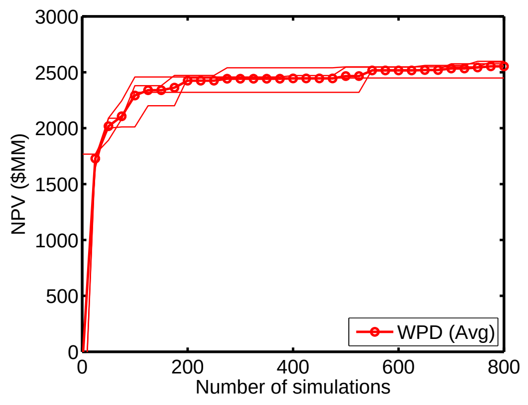
The four well pattern types shown in Fig. 3.1 were considered in the optimizations. We performed optimizations using one and four operators. For the optimizations that apply one operator per particle we used 20 PSO particles. In these optimizations, although each particle had only one pattern operation performed on it, the particular operator varied with particle and iteration. For the optimizations that apply four pattern operators per particle, we used 40 particles. More particles were used in the optimizations with four operators because the number of variables is about twice as many as in the optimizations with one operator. Four optimizations were performed in each case and each optimization was run for 40 iterations. Function evaluations were performed in parallel using a cluster of up to 40 processors.

Figures 3.10(a) and (b) show the evolution of the NPV of the best development scenario versus number of simulations for the optimizations using one and four operators. Each thin curve corresponds to a different optimization run and the heavy curve depicts the average of the best solutions from the four runs. NPV clearly improves with iteration, with the largest improvement coming at early iterations. Tables 3.3 and 3.4 summarize the results for the optimization runs with one and four operators. In the optimizations with one operator, the inverted nine-spot was the best pattern with an NPV of $2597 MM (Table 3.3). This development scenario has 30 producers and 9 injectors. For the optimizations with four operators, the inverted seven-spot pattern, containing 28 producers and 14 injectors, gave the best scenario with an NPV of $2872 MM (Table 3.4).

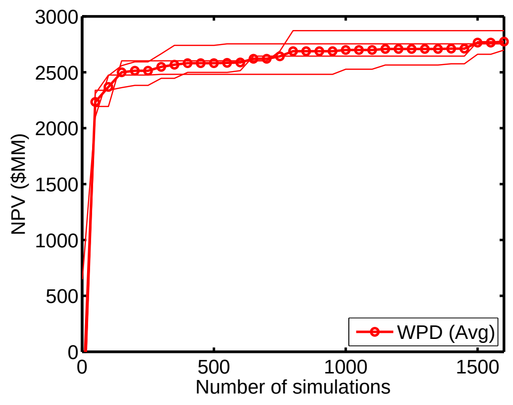
Table 3.3: Optimization results using one pattern operator (Example 1)

Run	Best pattern	NPV ($MM)	Well count	
			Producers	Injectors
1	inv. 9-spot	2591	26	10
2	inv. 7-spot	2449	28	14
3	inv. 6-spot	2575	28	9
4	inv. 9-spot	2597	30	9
Average		**2553**		

Although the results in Tables 3.3 and 3.4 suggest that the use of four operators provides generally better NPVs than those using one operator, it must be kept in mind that twice



(a) one pattern operator



(b) four pattern operators

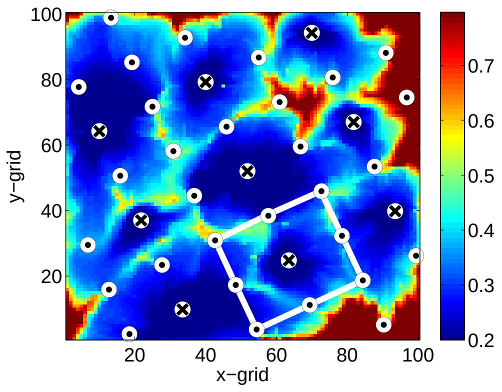
Figure 3.10: NPV of the best solutions versus number of simulations for the four optimization runs using one and four pattern operators (Example 1).

Table 3.4: Optimization results using four pattern operators (Example 1)

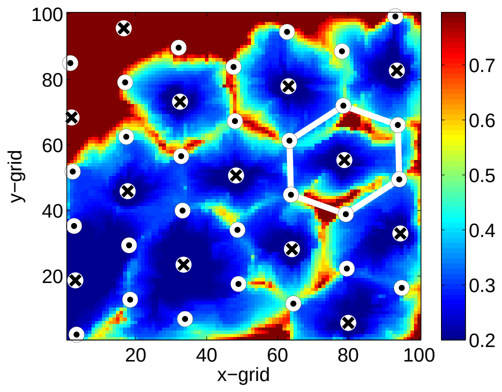
Run	Best pattern	NPV ($MM)	Well count	
			Producers	Injectors
1	inv. 9-spot	2754	30	9
2	inv. 7-spot	2872	28	14
3	inv. 9-spot	2698	33	9
4	inv. 9-spot	2773	28	8
Average		**2774**		

as many simulations were performed in the four-operator cases than in the one-operator cases. However, assessment of NPVs for the four-operator runs after 800 simulations indicates that on average these are superior to those from the one-operator runs after 800 simulations (average NPV of $2688 MM for the four-operator runs versus $2553 MM for the one-operator runs). The maximum NPV for the four-operator runs after 800 simulations ($2872 MM) was also considerably higher than that from the one-operator runs ($2597 MM). In all subsequent WPD optimizations described here, four pattern operators were used.

Figures 3.11(a) and (b) show the well locations from the best solutions in the optimizations with one and four operators. The black dots inside white circles represent producer wells while the black crosses inside white circles designate injection wells. In these figures, the basic pattern element is depicted by the white lines. The oil saturation at 500 days is shown as background. It is evident that the best patterns are, in both cases, rotated with respect to the reservoir boundaries. This results from the impact of reservoir heterogeneity on the flow field. We note that, based on the oil saturation maps, both patterns appear to provide efficient sweep. It is difficult to draw quantitative conclusions from these maps, however, as the pattern that appears to have more unswept oil at 500 days (Figure 3.11(b)) actually leads to a larger NPV over the full simulation.



(a) one pattern operator



(b) four pattern operators

Figure 3.11: Well locations of the best solutions using one operator and four operators (Example 1). Oil saturation at 500 days shown as background. The black dots inside white circles represent producer wells while while the black crosses inside white circles designate injection wells.

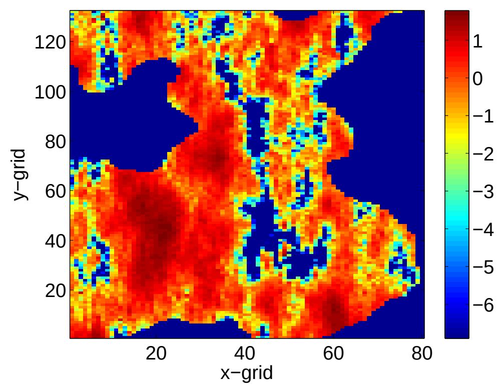
We next compared the optimization results with those obtained using standard well patterns (no optimization is performed). Results for standard patterns aligned with the reservoir boundaries are presented in Table 3.5. We considered well patterns with spacings from 20 acres to 50 acres. This range was determined based on the bounds specified for a and b and for the pattern operator parameters. Results for standard patterns with 20 acre spacings gave negative NPVs and are not presented in the table. It is clear that the optimization results are significantly better than those for the standard patterns, which highlights the potential benefits of using our procedure for large-scale optimizations.

Table 3.5: NPV of unoptimized standard well patterns for different well spacings. The well patterns are aligned with the reservoir boundaries (Example 1)

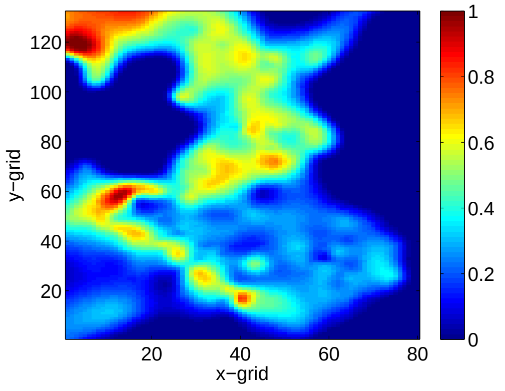
Spacing	Pattern	NPV ($MM)	Well count	
			Producers	Injectors
30 acres	inv. 5-spot	1432	25	25
	inv. 6-spot	749	25	45
	inv. 7-spot	-991	65	30
	inv. 9-spot	-1321	75	25
40 acres	inv. 5-spot	1895	16	16
	inv. 6-spot	1557	48	16
	inv. 7-spot	245	44	20
	inv. 9-spot	805	48	16
50 acres	inv. 5-spot	2151	16	16
	inv. 6-spot	1780	16	28
	inv. 7-spot	467	40	20
	inv. 9-spot	707	16	28

3.3.2 Example 2: WPD and WWP optimizations in reservoir with irregular boundary

Here, we applied the WPD and WWP procedures to maximize NPV in a reservoir with irregular boundaries. The simulator used for this case was 3DSL [83]. The two-dimensional synthetic reservoir model contains 80 $\times$ 132 grid blocks, with each block of dimensions 250 ft $\times$ 200 ft $\times$ 10 ft. Fig. 3.12(a) depicts the logarithm of the permeability field. Net-to-gross ratio varies from block to block, with blocks outside the boundary (feasible region) having zero net-to-gross (Fig. 3.12(b)). The reservoir initially contains oil, gas and water ($S_{oi} = 0.80$, $S_{gi} = 0.01$, $S_{wi} = 0.19$). For fluid viscosities, we specify $\mu_o = 1.2$ cp, $\mu_g = 0.01$ cp, and $\mu_w = 0.31$ cp. Relative permeability end points for oil and water are



(a) Logarithm of permeability field



(b) net-to-gross

Figure 3.12: Logarithm of permeability field and net-to-gross ratio (Example 2).

1.0 and 0.1 respectively. The initial reservoir pressure is 2700 psi. The production and injection wells operate under BHP constraints of 1200 psi and 2900 psi respectively. The total production time is 1825 days.

All of the pattern types shown in Fig. 3.1 were considered in the WPD optimizations. Each optimization run used 40 particles with four pattern operators applied. The runs proceed for 40 iterations. Five such WPD runs are performed. As is evident in Fig. 3.12, the region in which wells could be placed is irregular. In the WPD optimizations, well patterns were still replicated throughout the reservoir. Wells that fell outside the boundaries were eliminated from the total set, so only those wells located in the reservoir were included in the simulations.

Following the five WPD runs, we performed five WWP optimizations. The WWP optimizations were based on the best solution from the WPD runs (meaning the perturbations are computed with respect to the best configuration determined from the five WPD optimizations). In the WWP runs, the optimization parameters were defined such that the wells always fall within the feasible region after the perturbations.

Fig. 3.13 shows the evolution of the best solutions for the four optimization runs (thin lines) along with the average (solid line with circles). Table 3.6 presents the results for the five WPD optimizations. The results for the five runs were quite consistent, with the inverted five-spot found to be the best pattern in all cases. The maximum NPV for the five runs was \$1460 MM (run 3).

Using the best pattern from run 3, five WWP optimizations were then performed. The results are presented in Fig. 3.14, where the best WPD solution is shown (thick solid line, corresponding to the first 1600 simulations) along with the average of the five WWP runs (solid line with circles). It is clear that NPV increased in both phases (WPD and WWP) of the optimization and that WWP provides clear improvement over the WPD results. Results from all five WWP runs are shown in Fig. 3.15, where the NPV of the best scenario in each run (thin lines) is displayed along with the average curve. It is evident that all of the WWP runs provided an increase in NPV relative to the best WPD solution (dot-dash line). Results from the five WWP runs are summarized in Table 3.7. The maximum NPV is \$1801 MM, which represents an increase of \$341 MM (23.4%) over the best WPD result (run 3 in Table 3.6).

Figs. 3.16(a) and (b) show the well locations from the best WPD and WWP optimization runs. Although the perturbations evident in Fig. 3.16(b) do not appear that dramatic, this configuration resulted in a significant improvement in NPV over the unperturbed configuration.

We note finally that solving this optimization problem using a traditional approach, i.e., through use of concatenation of well parameters, would present some difficulties. For example, constraints must be introduced to keep wells within the feasible region and to satisfy minimum well-to-well distance requirements. Incorporation of these constraints into the optimization may limit the effectiveness of standard algorithms, particularly for large numbers of wells.

3.3.3 Example 3: WPD and WWP optimizations over multiple reservoir models

In this example, we accounted for geological uncertainty by performing the optimizations over five realizations of the reservoir model. In each phase of the optimization, we optimized expected NPV, designated $\langle \text{NPV} \rangle$, which is simply the average of the NPVs over the five models. The reservoir model consists of 63×63 blocks and the dimensions of each

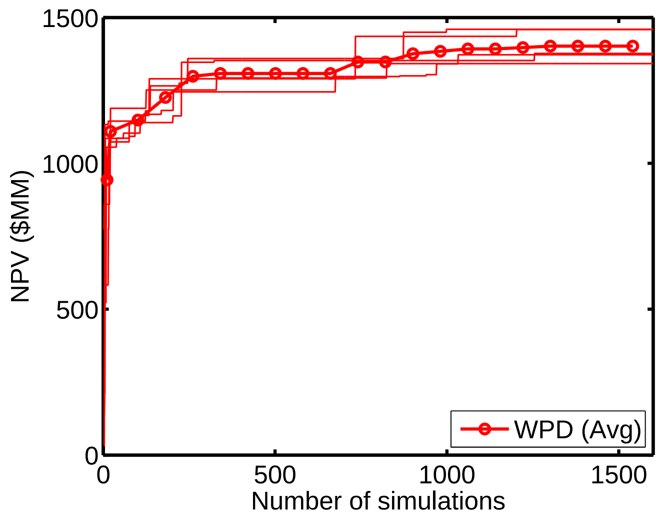


Figure 3.13: NPV of the best WPD solutions versus number of simulations (Example 2).

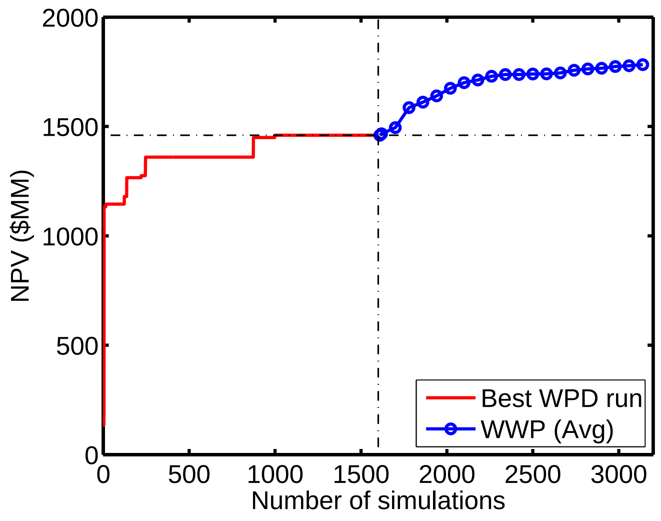


Figure 3.14: NPV of best result from WPD and average NPV of the best WWP solutions versus number of simulations (Example 2).

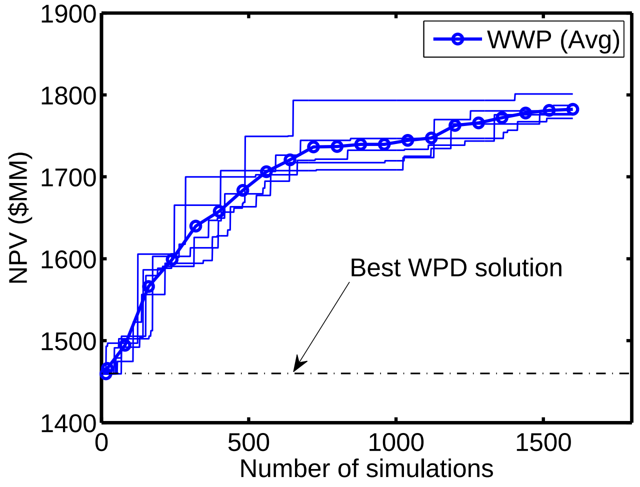


Figure 3.15: NPV of the best WWP solutions versus number of simulations (Example 2).

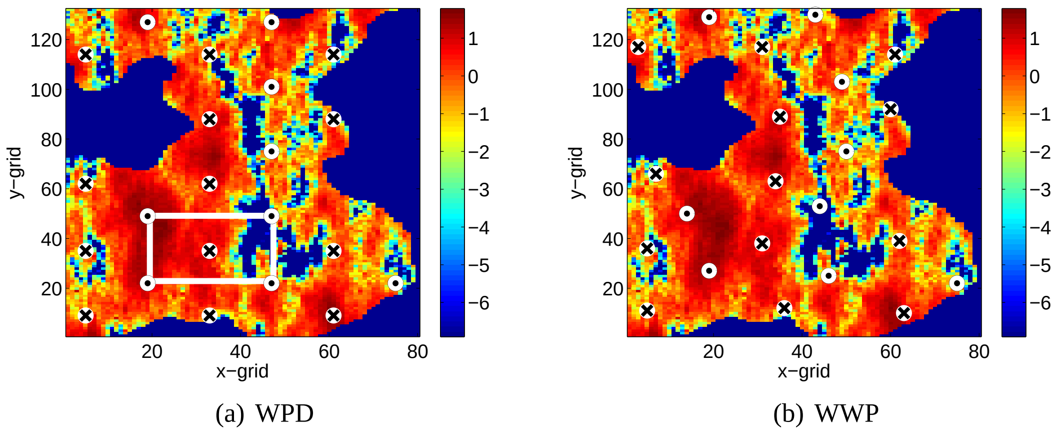


Figure 3.16: Well locations for the best WPD and WWP solutions. Logarithm of permeability field is shown as background (Example 2). The circles with black dots correspond to producers while the circles with crosses correspond to injectors.

Table 3.6: Optimization results using WPD with four pattern operators (Example 2)

Run	Best pattern	NPV	Well count	
		($MM)	Producers	Injectors
1	inv. 5-spot	1377	16	15
2	inv. 5-spot	1459	15	15
3	inv. 5-spot	1460	15	15
4	inv. 5-spot	1372	15	15
5	inv. 5-spot	1342	13	15
Average		**1402**		

Table 3.7: Optimization results using the WWP procedure (Example 2)

Run	NPV	Increase over WPD	
	($MM)	($MM)	%
1	1777	317	21.7
2	1787	327	22.4
3	1776	316	21.6
4	1801	341	23.4
5	1771	311	21.3
Average	**1782**	**322**	**22.1**

grid block are 100 ft $\times$ 100 ft $\times$ 30 ft. The permeability fields were generated geostatistically using an exponential variogram model with correlation length of 1000 ft. Porosity varies from block to block and is correlated with permeability.

We performed four WPD optimizations with four pattern operators. Using the best solution from the WPD runs, we then performed four WWP optimizations. Each optimization run contained 40 particles and was run for 40 iterations.

Table 3.8 shows the optimization results from the WPD runs. The best pattern is consistently a normal nine-spot pattern. Note that, although WPD only encodes the inverted forms of the well patterns, the optimization consistently switched from the inverted to the normal nine-spot. The best scenario (run 2) had an expected NPV of $832 MM and contained 7 producers and 29 injectors.

Fig. 3.17 shows the evolution of the best solutions in the four WPD optimizations (thin red lines) and the average ⟨NPV⟩ over all runs (solid red line with circles). Fig. 3.19 shows the evolution of ⟨NPV⟩ for the best scenarios in the four WWP optimizations (thin blue lines) along with the average result. There was very little variation between the four WWP optimizations. Again, ⟨NPV⟩ increased with iteration during both phases of the optimization. Fig. 3.18 shows the evolution of ⟨NPV⟩ in the best WPD run and the average ⟨NPV⟩ of the four subsequent WWP runs. The increase in ⟨NPV⟩ using WWP was substantial in the first few iterations. Table 3.9 summarizes the results of the WWP optimizations. The WWP procedure improved the NPV in each case, with a maximum increase in ⟨NPV⟩ of \$290 MM (34.9%) over the best WPD result.

Figs. 3.20(a) and (b) show the well locations for the best solutions from the two phases of the optimization, with the permeability field of one of the realizations shown in the background. The degree of perturbation of the well locations, evident in Fig. 3.20(b), was greater than that observed in Example 2.

Fig. 3.21 shows the ⟨NPV⟩s for each of the five realizations. We see that the use of WWP provided improvement for all realizations.

Table 3.8: Optimization results using WPD with four pattern operators (Example 3)

Run	Best pattern	⟨NPV⟩	Well count	
		(\$MM)	Producers	Injectors
1	norm. 9-spot	705	8	30
2	norm. 9-spot	832	7	29
3	norm. 9-spot	723	9	27
4	norm. 9-spot	757	7	30
Average		**754**		

3.3.4 Example 4: Comparison of WPD to well-by-well concatenation

In this example, we performed a limited comparison of optimizations using the WPD representation to optimizations using concatenated well parameters. We used the same reservoir model and economic parameters as in Example 1, except that the oil price was now \$80/bbl.

Table 3.9: Optimization results using the WWP procedure (Example 3)

Run	⟨NPV⟩	Increase over WPD	
	($MM)	($MM)	%
1	1116	284	34.1
2	1122	290	34.9
3	1119	287	34.5
4	1120	288	34.4
5	1115	283	34.0
Average	**1184**	**286**	**34.4**

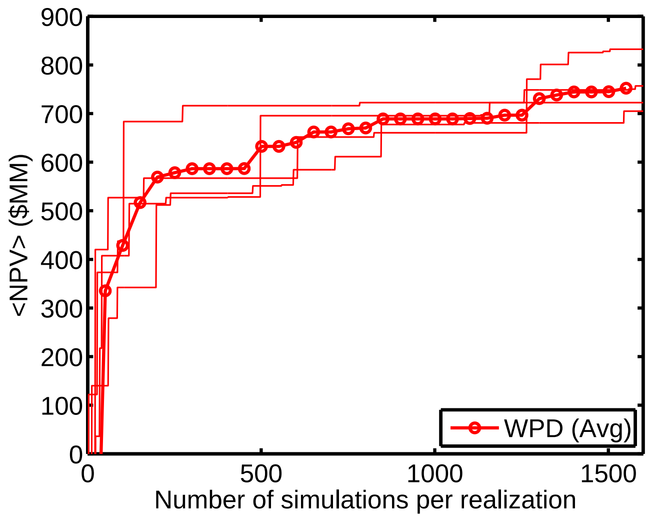


Figure 3.17: ⟨NPV⟩ of best results from the WPD optimizations versus number of simulations per realization (Example 3).

We first performed five WPD optimizations and then five optimizations with concatenated well parameters using the same number of wells as in the best WPD solution. In the optimizations using the well-by-well concatenation, we determined both well type and well locations. Well-to-well distance constraints were not included in the optimizations using concatenation. Other comparisons between the two techniques are also possible. The approach used here does not require the determination of the optimal number of wells in the

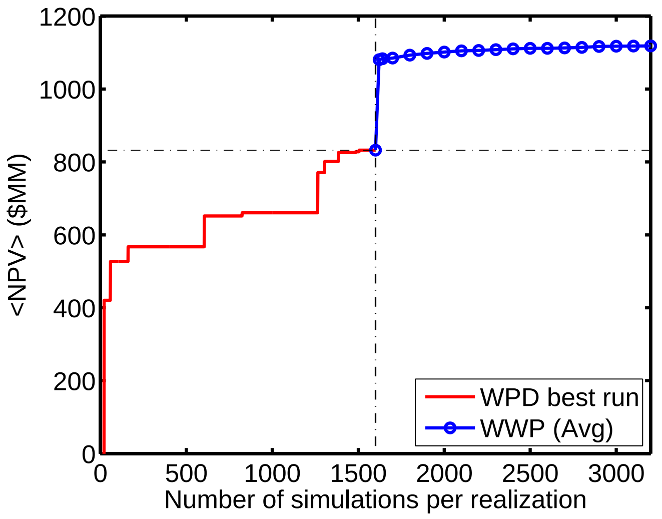


Figure 3.18: ⟨NPV⟩ of best result from WPD and average ⟨NPV⟩ of the best WWP solutions versus number of simulations per realization (Example 3).

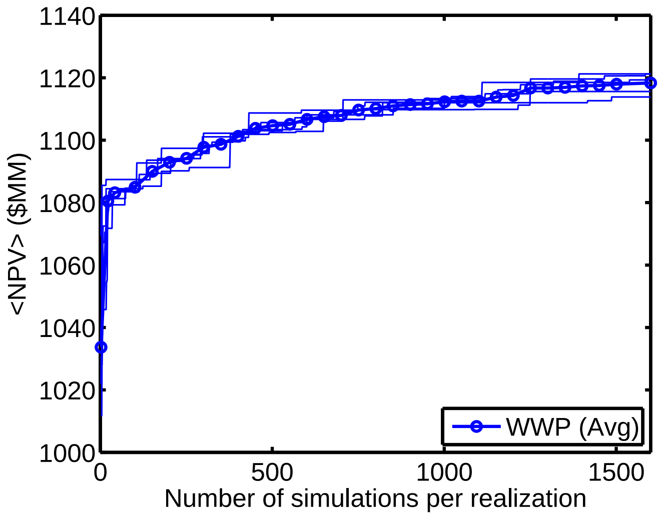


Figure 3.19: ⟨NPV⟩ of the best WWP solutions versus number of simulations per realization (Example 3).

well-by-well concatenation runs, so in this sense the problem setup provides an advantage to this approach.

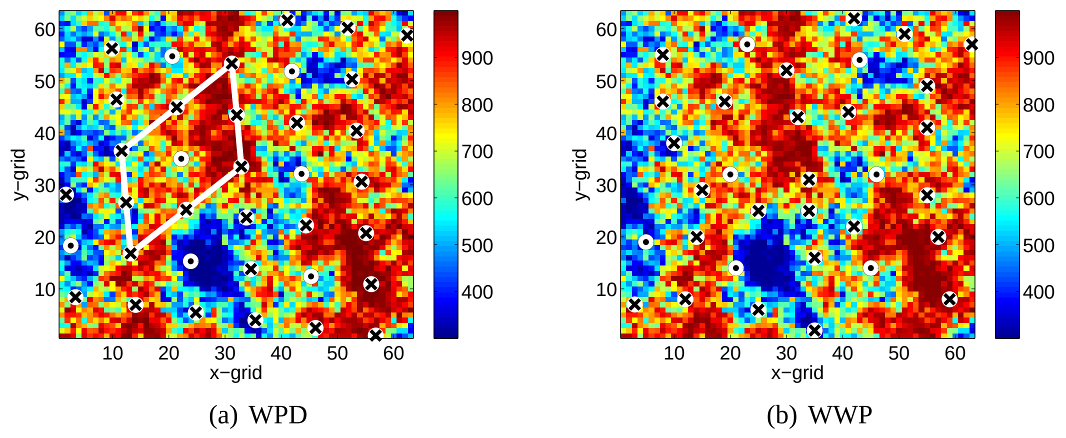


Figure 3.20: Well locations for the best WPD and WWP solutions. Permeability field for one of the realizations is shown as background (Example 3). The circles with black dots correspond to producers while the circles with crosses correspond to injectors.

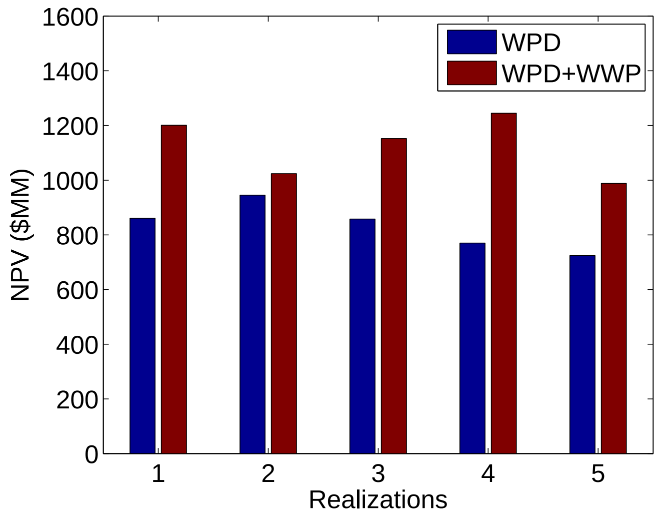


Figure 3.21: NPV of the phase 1 (WPD) and phase 2 (WWP) optimizations for each realization (Example 3).

In all optimizations, we used the PSO algorithm with 30 particles. Each optimization was run for 15 iterations. Fig. 3.22 shows the evolution of the average NPV of the best solutions from the five WPD runs and the analogous result for the runs using the concatenated approach. The WPD approach clearly outperformed the well-by-well concatenation. This demonstrates that WPD indeed provides a useful and concise solution representation. It is also worth noting that the optimized scenarios determined using well-by-well concatenation did not satisfy well-to-well distance constraints, while those using WPD did satisfy these constraints. We note finally that further improvement of the WPD results could be achieved through use of WWP.

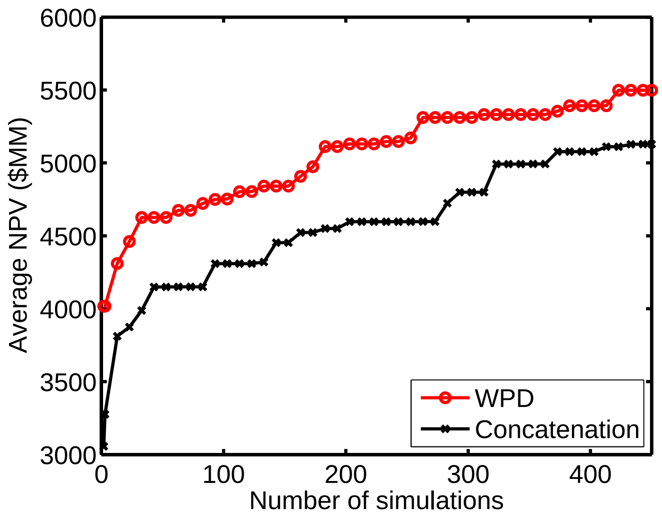


Figure 3.22: Comparison of the average NPV of the best solutions versus number of simulations for optimizations using WPD and well-by-well concatenation (Example 4).

3.4 Summary

In this chapter, we described the development and application of a new well pattern optimization (WPO) algorithm for optimizing well locations in large-scale field development. The procedure comprises a well pattern description and a well-by-well perturbation procedure. All optimizations are performed using the PSO algorithm. We considered different optimization problems and different variants of the WPO procedure. In the examples considered, the WPD procedure provided significant increases in NPV, and further improvements are possible using the WWP technique. These examples demonstrate the applicability of the WPO procedure for optimization problems with many wells.

In the WPO optimizations, we have used default PSO parameters ($\omega,\ c_1,\ c_2$) and the random variable neighborhood topology described in Chapter 2. Better optimization results may be obtained by tuning the PSO parameters. In the next chapter, we describe the application of a metaoptimization procedure for optimizing PSO parameters in the WPO optimizations.

Chapter 4

Metaoptimization for PSO Parameter Determination

In this chapter, we describe the implementation of metaoptimization techniques to determine optimal PSO parameters. First, we describe the metaoptimization procedure. We applied this approach to determine appropriate parameter values by considering four benchmark well placement optimization problems. The optimized PSO parameters were then used for two realistic well placement optimization problems including optimizing the location of 15 vertical wells and WPD optimizations in two-dimensional reservoir models. For these problems, we performed several optimizations using default and optimized parameters. We also applied the metaoptimization procedure directly to the target optimization problems (i.e., the optimal parameters are determined for the actual problem, not the benchmark problems). The results in this chapter demonstrate the effectiveness of the metaoptimization procedure.

4.1 PSO Metaoptimization

In Chapters 2 and 3 we applied the PSO algorithm for well placement optimization, using PSO parameters taken from the literature. As is the case for other stochastic optimization algorithms, the performance of PSO depends on the values assigned to the algorithm parameters. In this chapter, we describe the application of a metaoptimization procedure to optimize the key PSO parameters directly.

In PSO metaoptimization, two PSO algorithms are used [2, 69]. The first PSO algorithm optimizes PSO parameters, while the second PSO algorithm uses the optimized parameters to solve a specified optimization problem or problems. The first and second PSO algorithms are referred to as the "superswarm PSO" and "subswarm PSO" respectively [69]. The metaoptimization procedure can be used to determine optimal PSO parameter values for a set of small benchmark problems or it can be directly applied to the target optimization problem. In the former case, the assumption is that the parameters that are optimal for the benchmark problems are also appropriate for other problems. We consider both approaches in this chapter.

The metaoptimization procedure consists of three components: superswarm PSO, subswarm PSO, and one or more optimization problems (optimized by the subswarm PSO). Each superswarm particle has associated PSO subswarms. Several subswarm optimizations are performed in order to evaluate the objective function value of a superswarm particle. The relationship between the superswarm and subswarm PSO algorithms is shown in Fig. 4.1. In the figure, N designates the number of subswarm particles and T the number of subswarm iterations. We now describe the superswarm and subswarm PSO procedures.

4.1.1 Superswarm PSO

The superswarm PSO optimizes the parameters required by the PSO algorithm. As shown in Eq. 2.2, repeated here for completeness, some of these parameters (ω, c_1, c_2) are used in the computation of PSO particle velocity, $\mathbf{v}_i(k+1)$:

$$\begin{aligned}\mathbf{v}_i(k+1) = {} & \omega \cdot \mathbf{v}_i(k) \\ & + \ c_1 \cdot \mathbf{D}_1(k) \cdot (\mathbf{x}_i^{pbest}(k) - \mathbf{x}_i(k)) \\ & + \ c_2 \cdot \mathbf{D}_2(k) \cdot (\mathbf{x}_i^{nbest}(k) - \mathbf{x}_i(k)),\end{aligned} \tag{4.1}$$

where ω, c_1 and c_2 are the weights of the inertia, cognitive, and social components; $\mathbf{D}_1(k)$ and $\mathbf{D}_2(k)$ are diagonal matrices whose diagonal components are uniformly distributed random variables in the range [0, 1]; $\mathbf{x}_i(k)$ is the position of the ith particle at iteration k; $\mathbf{x}_i^{pbest}(k)$ is the previous best particle position, and $\mathbf{x}_i^{nbest}(k)$ is the best position in the

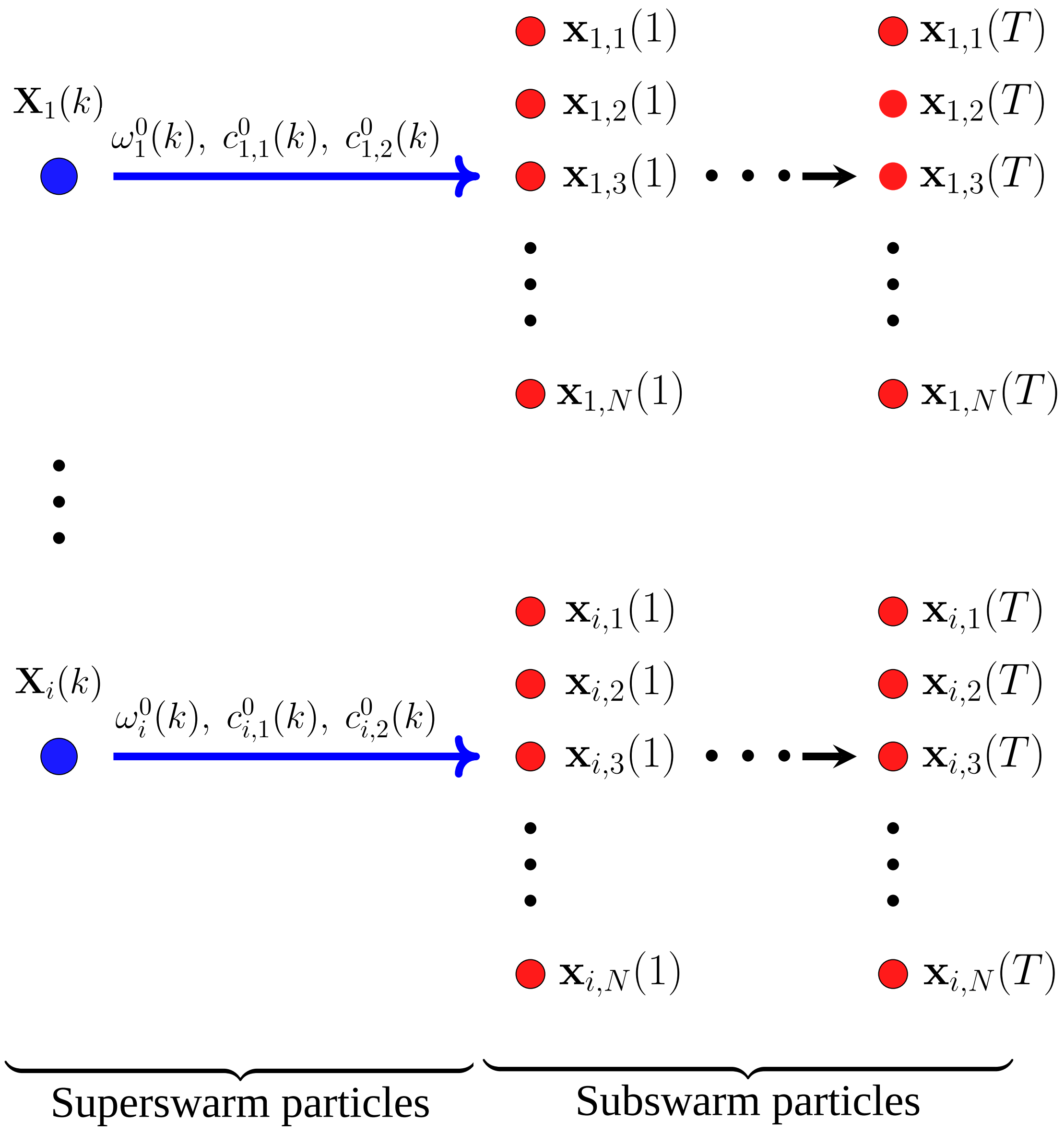


Figure 4.1: Illustration of the PSO metaoptimization procedure for one iteration of the superswarm PSO.

neighborhood of particle i. In addition to these parameters, we can also optimize the neighborhood topology type (n_{type}). The topology determines how $\mathbf{x}^{nbest}$ is determined. In our optimizations, we will consider four different neighborhood topologies including the star, ring, cluster and random variable topologies. These topologies are described in Section 2.1.1.

The superswarm PSO seeks to minimize average error (if the global optimum is known) or maximize the average objective function values from several subswarm PSO optimizations. Multiple subswarm optimizations are performed (using the same PSO parameters) because of the stochastic nature of the PSO algorithm.

Each superswarm particle represents the set of PSO parameters to be optimized. The definition of the objective function of each superswarm particle depends on the number of test optimization problems considered and on the number of repetitions of the subswarm optimizations. We now describe the objective function evaluation for the superswarm PSO where we seek to determine optimal parameter values for ω, c_1, c_2 and n_{type}. Let $p,\ p \in \{1,\ 2, \ldots, P\}$ be the index of a test optimization problem. Let $\mathbf{X}_i(k)$ be the position of the ith superswarm particle at iteration k and let $e_{p,r}$ denote the best solution obtained from run r of a subswarm optimization using test problem p. The objective function value of the ith particle, $F_i(\mathbf{X}_i(k))$, is computed as follows:

$$F_i(\mathbf{X}_i(k)) = \frac{1}{P}\sum_{p=1}^{P}\frac{1}{R}\sum_{r=1}^{R} e_{p,r}\ , \tag{4.2}$$

where P is the number of test optimization problems considered and R is the number of repetitions of the subswarm optimizations. In our applications we used values of p from 1 to 4 and for the metaoptimizations using benchmark problems we specified $R = 20$.

4.1.2 Subswarm PSO

The subswarm PSO optimizes the test problem(s) using the parameters from the associated superswarm particle. Let $\mathbf{x}_{i,j}(t)$ denote the position of the jth subswarm associated with the ith superswarm particle, $\mathbf{X}_i(k)$ (see Fig. 4.1). In each iteration of the superswarm PSO, evaluation of the objective function for each superswarm particle requires one or more subswarm optimizations. For each subswarm PSO, the PSO parameters are fixed. In the subswarm PSO, the particle velocity, $\mathbf{v}_{i,j}(t+1)$, is computed as follows:

$$\begin{aligned}\mathbf{v}_{i,j}(t+1) = {} & \omega_i^0(k)\cdot\mathbf{v}_{i,j}(t) \\ & +\ c_{i,1}^0(k)\cdot\mathbf{D}_1(t)\cdot(\mathbf{x}_{i,j}^{pbest}(t)-\mathbf{x}_{i,j}(t)) \\ & +\ c_{i,2}^0(k)\cdot\mathbf{D}_2(t)\cdot(\mathbf{x}_{i,j}^{nbest}(t)-\mathbf{x}_{i,j}(t)),\end{aligned} \tag{4.3}$$

where t is the index of subswarm iteration and $\omega_i^0(k)$, $c_{i,1}^0(k)$ and $c_{i,2}^0(k)$ are the parameter values from superswarm particle $\mathbf{X}_i(k)$. The objective function of a subswarm particle depends on the optimization problem to be solved. If the global optimum is known, then the subswarm PSO seeks to minimize error. Otherwise, the fitness of the actual optimization problem is maximized (or minimized). The overall flow chart of the metaoptimization procedure is shown in Fig. 4.2.

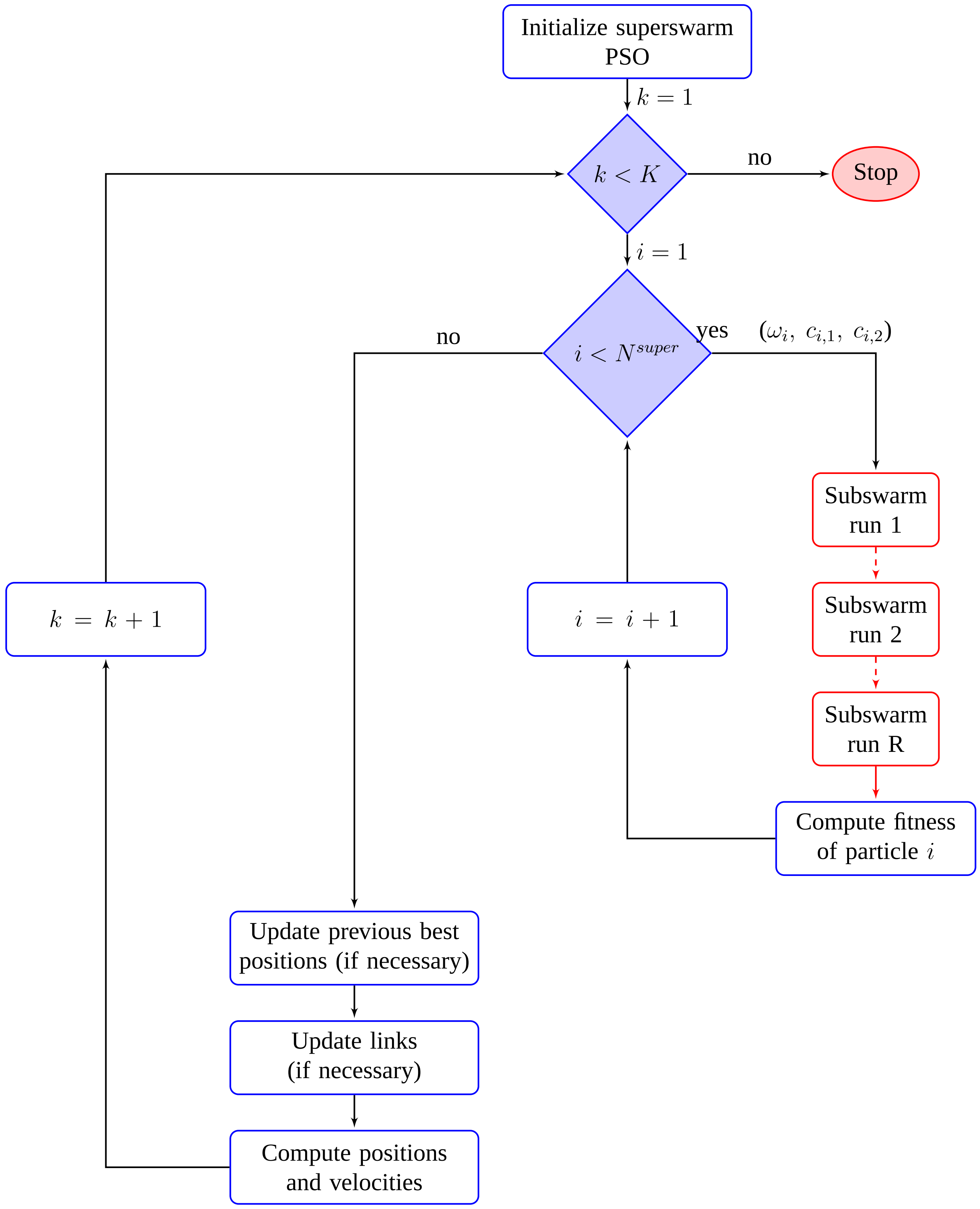

Initialize superswarm PSO
$k = 1$
$k < K$
no
Stop
$i = 1$
$i < N^{super}$
no
yes
$(\omega_i,\ c_{i,1},\ c_{i,2})$
Subswarm run 1
Subswarm run 2
Subswarm run R
Compute fitness of particle i
$k = k + 1$
$i = i + 1$
Update previous best positions (if necessary)
Update links (if necessary)
Compute positions and velocities

Figure 4.2: Flowchart of the overall metaoptimization procedure.

4.2 Benchmark Well Placement Optimization Problems

In the metaoptimization procedure, one or more optimization problems are used to evaluate the subswarm. We considered four well placement problems as benchmark problems. In all problems, we maximized NPV by optimizing the location of one or two wells. Because of the low dimension of the search space and the relatively small model sizes considered, we were able to compute (and save) the full objective function exhaustively for each problem. This allowed us to perform function evaluations by table look-up and enabled inexpensive repetition of the optimization runs.

Benchmark problems 1, 3, and 4 use the same reservoir model. In problem 2, we maximized expected NPV by optimizing the location of a single producer over ten realizations of a reservoir model containing 40$\times$40 grid blocks. The details of the reservoir model and problem set up for problem 2 were described in Section 2.3.1 and are not repeated here.

Reservoir model for problems 1, 3, and 4

In problems 1, 3, and 4, we used a synthetic, heterogeneous, two-dimensional reservoir model containing 20$\times$20 grid blocks. Each grid block is 150 ft$\times$150 ft$\times$50 ft. The permeability field is shown in Fig. 4.3. Porosity is 0.20 and assumed constant in all grid blocks. The reservoir initially contains oil and water ($S_{oi} = 0.85$, $S_{wi} = 0.15$). The oil viscosity μ_o is 1.20 cp and oil compressibility c_o is 2.0$\times 10^{-5}$ psi^{-1}. For water we specify μ_w = 0.31 cp and c_w = 2.9$\times 10^{-6}$ psi^{-1}. Relative permeability end points for oil and water are 0.85 and 0.30 respectively. The initial reservoir pressure is 4800 psi. Producer wells operate under BHP constraints of 1000 psi, while the injector (in problem 4) operates at 5000 psi. Total production time is 1500 days. For the NPV computation, we use an oil price of 50 $/bbl, water production and injection costs of 5 $/bbl, well cost of 20\times 10^6$, and a discount rate of 10%.

Benchmark problem set up

We varied the number and type of wells in the benchmark problems. A single producer was considered in problems 1 and 2, two producers in problem 3, and a producer and an injector in problem 4. The problems are summarized in Table 4.1.

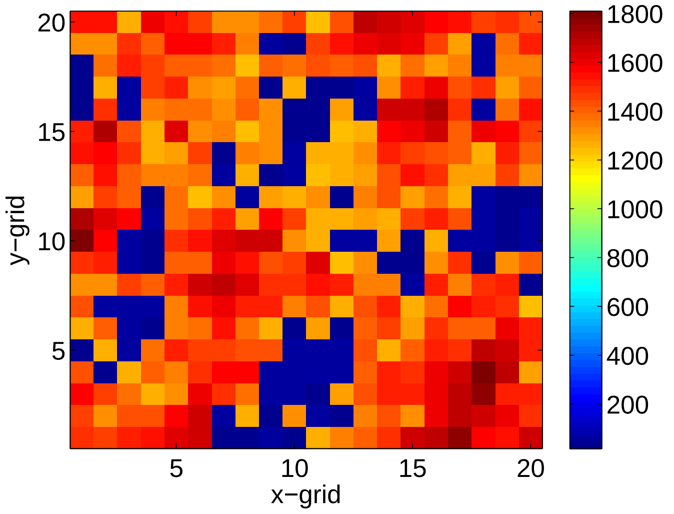


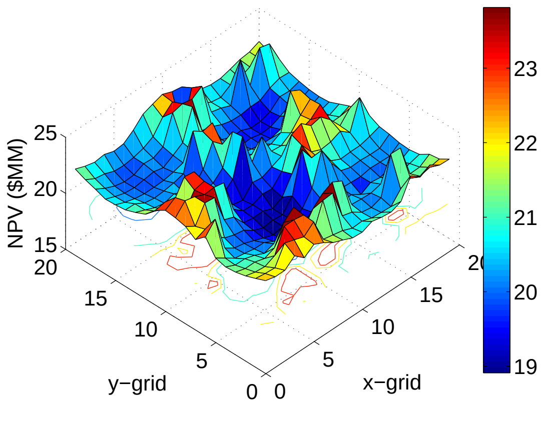
Figure 4.3: Permeability field for benchmark problems 1, 3, and 4.

Table 4.1: Benchmark optimization problems

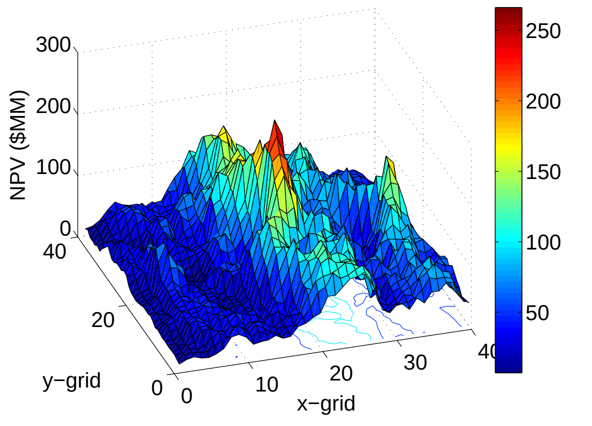
Problem	Reservoir model	Well count		# of opt. variables	# of feasible solutions
		Prod.	Inj.		
1	$20 \times 20 \times 1$	1	0	2	400
2	$40 \times 40 \times 1$	1	0	2	1,600
3	$20 \times 20 \times 1$	2	0	4	159,600
4	$20 \times 20 \times 1$	1	1	4	159,600

As mentioned previously, we sampled the search space of each problem exhaustively. The number of function evaluations required for each problem is indicated in Table 4.1. In problems 3 and 4, there is the possibility of invalid solutions (two wells located in the same grid block). We penalized invalid solutions by assigning a large negative NPV value. All simulations were performed using Stanford's General Purpose Research Simulator (GPRS) [77, 78].

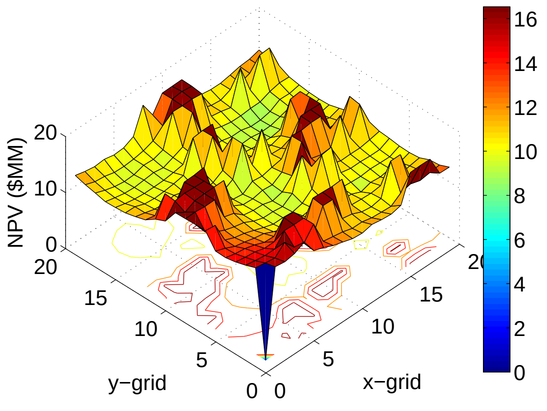
Fig. 4.4 shows the objective function surface for each optimization problem. The surfaces for problems 3 and 4 were obtained by taking a slice of the full solution with the second well at grid block (1,1). These surfaces are, in general, rough and discontinuous. The global optimum values for problems 1, 2, 3, and 4 are 23.81 \$MM, 266.18 \$MM, 19.43 \$MM, and 279.49 \$MM respectively.



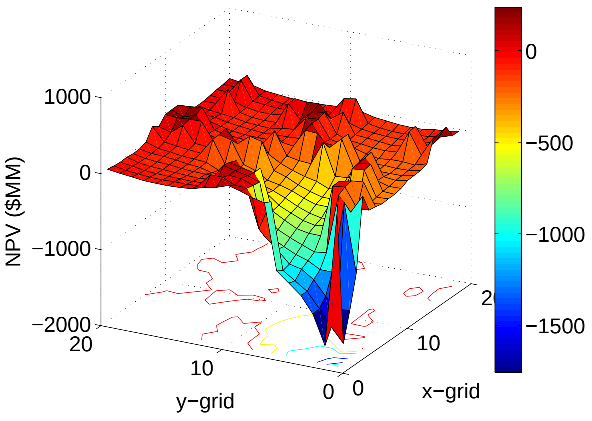
(a) Problem 1



(b) Problem 2



(c) Problem 3



(d) Problem 4

Figure 4.4: Objective function surfaces for the four optimization problems considered.

.

4.3 Metaoptimization Using the Benchmark Problems

We applied the metaoptimization for PSO parameter determination. Two metaoptimization cases were considered. In case 1, we optimized ω, c_1, and c_2 in the superswarm. In these optimizations, we used the random variable neighborhood topology with the number of particles informed set at 3 (these are the default topology parameters used in Chapters 2 and 3). In case 2, we included the neighborhood type in the metaoptimization, i.e., we optimized ω, c_1, c_2, and n_{type}. For this case, four neighborhood topologies were considered: star, ring, cluster, and random variable. The parameter ranges for the optimization variables are shown in Table 4.2. The ranges for ω, c_1, and c_2 are based on previous studies on the stability and convergence of PSO particle trajectories [49, 50]. All of the optimized parameters are continuous except for n_{type} which is discrete.

Table 4.2: Optimized PSO parameters

Parameter	Range		Type
	Minimum	Maximum	
ω	0	1	continuous
c_1	0	2.0	continuous
c_2	0	2.0	continuous
n_{type}	1	4	discrete

In each metaoptimization case, we performed optimizations first using the test problems separately, and then using all four together. For all optimizations, we set the superswarm PSO population size $N^{super} = 20$, maximum iteration, K, to 100, $\omega = 0.721$, $c_1 = c_2 = 1.193$, and used the random variable topology.

For the subswarm PSO, we also used a population size $N^{sub} = 20$. The number of iterations, T, in the subswarm PSO optimization runs depends on the test problem considered. In problem 1, $T = 400$; in problem 2, $T = 1600$ (per realization); and in problems 3 and 4, $T = 2000$. The T values for problems 1 and 2 are based on the number of function evaluations required to find the optimum solution by exhaustive search. The number of repetitions, R, of the subswarm runs was set to 20. We performed five superswarm optimization runs.

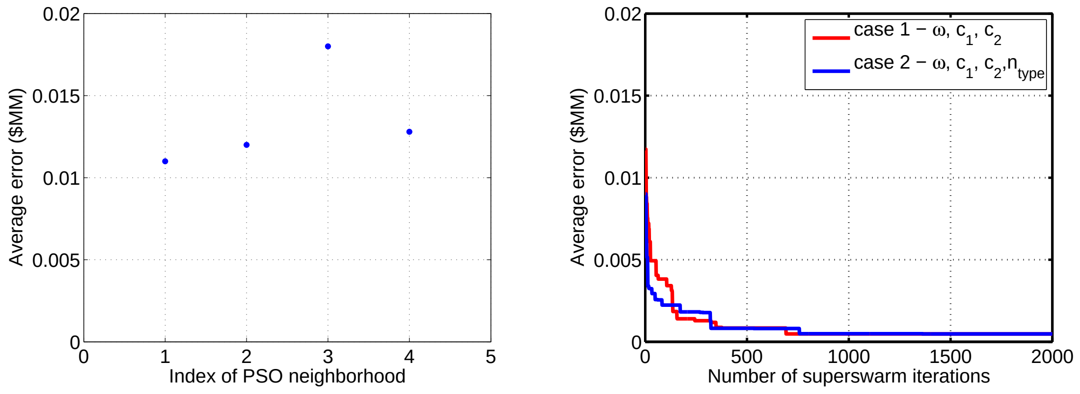
We also performed PSO optimizations for each benchmark problem using the default parameter values ($\omega = 0.721$, $c_1 = c_2 = 1.193$). Optimizations were performed using this parameter set and the four different neighborhood topologies described earlier. For each problem, we repeated the optimizations 20 times using a population size of 20 and number of iterations (T) as defined in the subswarm optimizations above.

Metaoptimization results

Figures 4.5-4.8 show the optimization results using standard PSO with unoptimized parameters (plots on left) and metaoptimization (plots on right) for problems 1-4. The metaoptimization results were averaged over five runs. The results for PSO with unoptimized parameters applied the default parameter set and the four different neighborhood types considered. The numbers on the x-axis of the plot represent the index of the neighborhood topology (1 - star, 2 - ring, 3 - cluster, 4 - random) used in the optimizations. The results show that metaoptimization provided better results, on average, compared to PSO with unoptimized parameters. This means that the metaoptimization procedure was able to find combinations of $\omega,\ c_1$, and c_2 that result in improved algorithm performance. The metaoptimization results when all four benchmark problems were optimized in the subswarm optimizations are shown Fig. 4.9.

The optimized parameter values for the benchmark problems, when considered separately, are shown in Tables 4.3-4.6. The metaoptimization results when all benchmark problems were considered together are shown in Table 4.7. In general, the optimized parameter values varied with the benchmark problem. They also depended on whether or not the neighborhood topology type was optimized. The weights of the cognitive and social component of the velocity equation (c_1, c_2) showed some relationship. In almost all cases, the optimized values of c_2 were larger than those of c_1 which indicates a preference for moving towards the neighborhood best solution at each iteration. Figure 4.10 displays the histogram of the optimized neighborhood topology type for the case 2 metaoptimization runs. The preferred topology type in 23 of the 25 runs was a topology with multiple neighborhoods (i.e., not the star topology), with the ring and cluster topologies selected the most frequently.

In summary, the metaoptimization results demonstrated that the superswarm is able to find parameters that provided better performance compared to using PSO with unoptimized (standard) parameters. However, the optimum values of ω, c_1, and c_2 varied with the benchmark problem considered, and whether or not the neighborhood topology type was optimized. In the next section, we will describe the application of the optimized parameter values and a metaoptimization procedure for realistic well placement optimization problems.



(a) Base case (b) Metaoptimization

Figure 4.5: Comparison of metaoptimization and standard PSO (with different topologies) results for problem 1.

4.4 Applications of Metaoptimization

We applied the metaoptimization procedure and optimized parameters to more realistic well placement problems. We considered two optimization problems: optimizing the location of 15 vertical wells and WPD optimizations in a two-dimensional reservoir. For each problem, we performed optimizations using PSO with default and optimized parameters. In the optimizations that used optimized parameters, we selected the parameters from the best case 1 and 2 metaoptimization runs when all the benchmark problems were considered together (Table 4.7). The default and optimized PSO parameters are shown in Table 4.8. We also

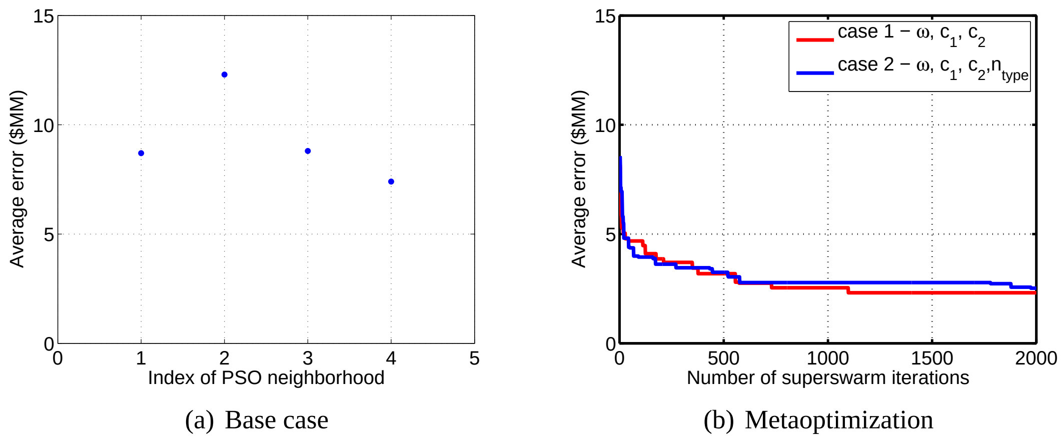


Figure 4.6: Comparison of metaoptimization and standard PSO (with different topologies) results for problem 2.

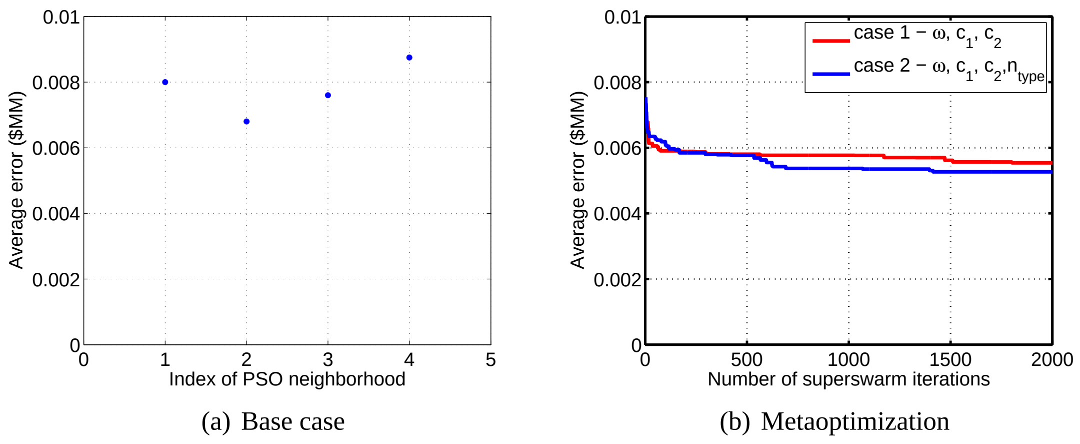


Figure 4.7: Comparison of metaoptimization and standard PSO (with different topologies) results for problem 3.

applied the metaoptimization procedure directly to the two target optimization problems. The economic parameters used for the computation of NPV are shown in Table 4.9.

4.4.1 Example 1: Optimizing the location of 15 vertical wells

In this example, we maximized NPV by optimizing the location of 15 vertical wells consisting of ten producers and five injectors. We used a synthetic, heterogeneous two-dimensional

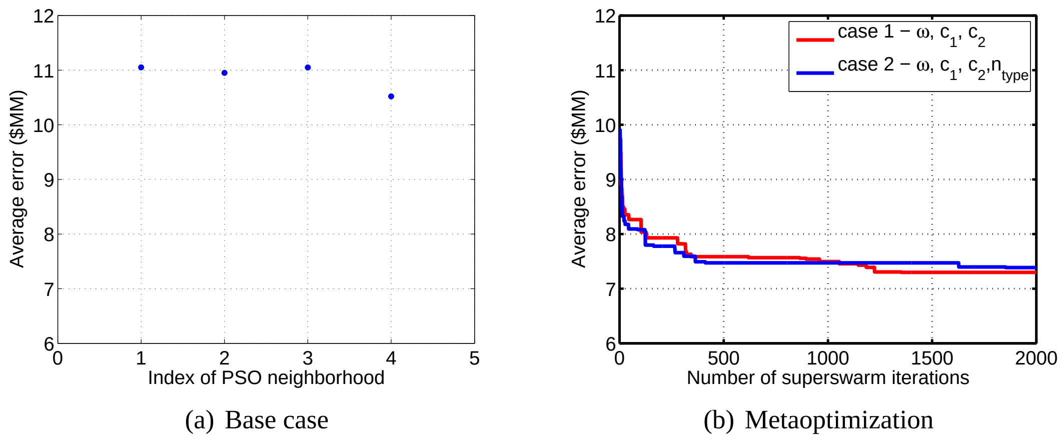


Figure 4.8: Comparison of metaoptimization and standard PSO (with different topologies) results for problem 4.

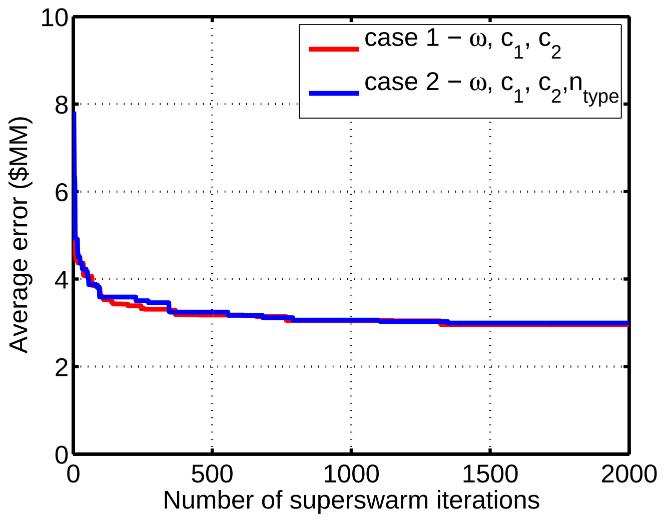


Figure 4.9: Metaoptimization results for problems 1-4.

reservoir model containing 40×40 grid blocks, with each grid block of dimensions 150 ft×150 ft×50 ft. The permeability field for the reservoir model is shown in Fig. 4.11. The initial reservoir pressure is 4800 psi and the producers and injectors operate under BHP constraints of 2000 psi and 5000 psi. Total production time is 1825 days. The other reservoir and fluid properties are the same as those used in Section 3.3.1. In the optimizations

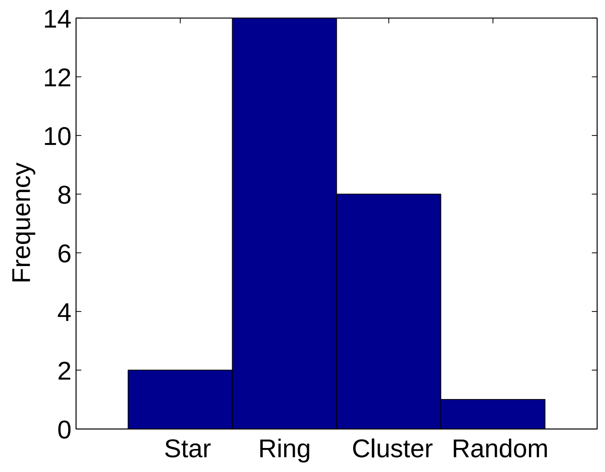


Figure 4.10: Distribution of neighborhood topology type from the metaoptimization runs using the benchmark optimization problems.

Table 4.3: Metaoptimization results using only problem 1

Run	case 1				case 2				
	Avg. Err ($MM)	ω	c_1	c_2	Avg. Err ($MM)	ω	c_1	c_2	n_{type}
1	4.25e-04	0.565	0.208	1.886	6.80e-04	0.203	0.374	1.999	2
2	3.40e-04	0.147	0.393	1.891	3.40e-04	0.095	0.548	1.958	3
3	7.65e-04	0.658	0.422	1.825	5.10e-04	0.154	1.177	1.748	1
4	4.25e-04	0.097	0.548	1.745	4.25e-04	0.425	0.490	1.857	3
5	4.25e-04	0.576	1.309	1.414	4.25e-04	0.549	0.295	1.981	2
Average	**4.76e-04**				**4.76e-04**				

performed here, the well-by-well parameters were concatenated, resulting in a total of 30 variables (two per well). However, we imposed a minimum well-pair distance constraint of 750 ft (five grid blocks) in order to obtain realistic solutions and eliminate solutions where wells may be too close. All infeasible well configurations (i.e., those that violate the well-pair distance constraint) were penalized by assigning a negative NPV.

For the optimizations using the default and optimized parameters, we used a swarm

Table 4.4: Metaoptimization results using only problem 2

Run	case 1				case 2				
	Avg. Err ($MM)	ω	c_1	c_2	Avg. Err ($MM)	ω	c_1	c_2	n_{type}
1	2.10	0.360	0.291	1.638	2.10	0.063	1.055	1.937	2
2	2.10	0.258	0.043	1.432	3.15	0.409	0.250	1.975	3
3	3.15	0.299	1.180	1.589	2.10	0.012	0.621	1.665	2
4	2.10	0.353	0.568	1.485	3.15	0.493	0.404	1.667	4
5	2.10	0.040	0.743	1.987	2.10	0.392	1.037	1.706	3
Average	**2.31**				**2.52**				

Table 4.5: Metaoptimization results using only problem 3

Run	case 1				case 2				
	Avg. Err ($MM)	ω	c_1	c_2	Avg. Err ($MM)	ω	c_1	c_2	n_{type}
1	5.28e-03	0.419	0.700	1.783	5.27e-03	0.304	0.063	1.746	3
2	5.64e-03	0.872	1.242	0.006	5.34e-03	0.824	0.357	1.233	3
3	5.43e-03	0.635	1.403	1.391	5.18e-03	0.260	0.106	1.854	2
4	5.60e-03	0.556	0.119	1.811	5.43e-03	0.015	0.782	1.523	3
5	5.78e-03	0.006	1.612	0.528	5.11e-03	0.018	0.463	1.834	2
Average	**5.55e-03**				**5.27e-03**				

size of 40. We used a large swarm size in this case because there were many invalid solutions resulting from the well-pair distance constraint. The stopping criterion for the optimizations was based on the number of simulations performed for feasible solutions. We set this number at 1000, i.e., we ran the optimizations until 1000 simulations of feasible scenarios were performed. We performed five optimization runs for each problem.

For the metaoptimization runs, we reduced the swarm size and number of iterations such that the number of simulations was reasonable. In the superswarm, we used a swarm size of 10 and ran the optimizations for five iterations. For the associated subswarm optimizations, we used 20 particles and ran the optimizations for 100 simulations. To reduce

Table 4.6: Metaoptimization results using only problem 4

Run	case 1				case 2				
	Avg. Err ($MM)	ω	c_1	c_2	Avg. Err ($MM)	ω	c_1	c_2	n_{type}
1	7.35	0.110	1.899	1.827	7.69	0.687	0.005	1.419	2
2	7.43	0.071	0.141	1.764	7.09	0.323	0.187	1.934	3
3	7.24	0.094	1.340	1.872	7.20	0.413	1.867	1.762	2
4	7.03	0.343	0.263	1.368	7.63	0.084	0.241	1.690	1
5	7.45	0.993	0.032	1.692	7.33	0.220	0.029	1.875	2
Average	**7.30**				**7.34**				

Table 4.7: Metaoptimization results using all benchmark problems

Run	case 1				case 2				
	Avg. Err ($MM)	ω	c_1	c_2	Avg. Err ($MM)	ω	c_1	c_2	n_{type}
1	2.58	0.493	0.665	1.467	2.86	0.970	1.152	1.950	2
2	2.86	0.066	0.552	1.781	2.84	0.819	0.178	1.446	2
3	3.57	0.254	1.438	1.452	3.47	0.986	1.226	1.911	2
4	3.08	0.676	0.072	1.919	3.06	0.010	0.858	1.487	1
5	2.70	0.368	1.318	1.562	2.73	0.285	0.639	1.805	2
Average	**2.96**				**2.99**				

Table 4.8: Default and optimized PSO parameters

Parameter	Default	Optimized Parameters	
		case 1	case 2
ω	0.721	0.493	0.285
c_1	1.193	0.665	0.639
c_2	1.193	1.467	1.805
n_{type}	Random	Random	Ring

Table 4.9: Economic parameters for NPV computation

Well cost	3×10^6 (\$)
Oil price	80 (\$/STB)
Water production cost	10 (\$/STB)
Water injection cost	10 (\$/STB)
Discount rate, r	0.10

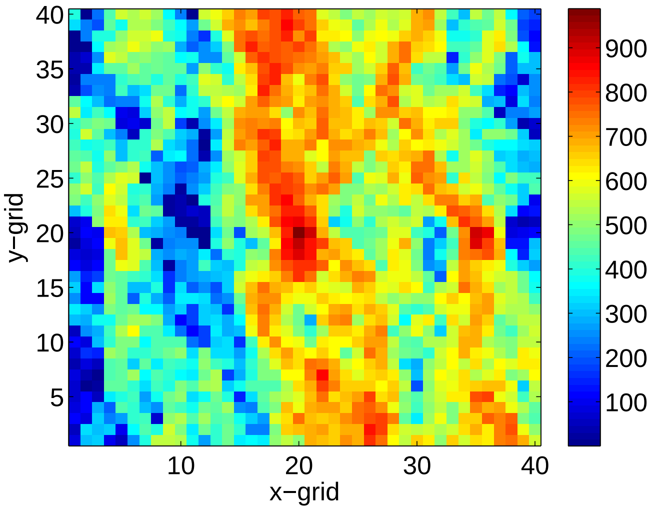


Figure 4.11: Permeability field for Example 1.

computational demands, we did not repeat the subswarm optimizations (i.e., $R = 1$). We performed five metaoptimization runs. All metaoptimization runs were stopped when the number of feasible simulations reached 1000.

Fig. 4.12 shows the average NPV of the best solutions versus the number of simulations performed. The metaoptimization results (cases 1 and 2) were better than those obtained from the optimizations using the default and optimized parameters. In Fig. 4.12, the case 1 metaoptimization (where we optimized $\omega,\ c_1,\ c_2$) shows a rapid increase in average NPV in early iterations. Metaoptimization with fixed neighborhood topology (case 1) provided (slightly) better results compared to the case where the neighborhood topology was optimized. The results for the optimizations using optimized parameters do not show consistent advantages over the default parameters. In case 1, these parameters provided better results, but in case 2 they did not. Table 4.10 summarizes the average NPV for the different optimizations and Table 4.11 gives the optimized parameters. The well locations from the best optimization runs are shown in Fig. 4.13. In all cases, the well locations are at least five grid blocks apart, although they do not exhibit any clear pattern.

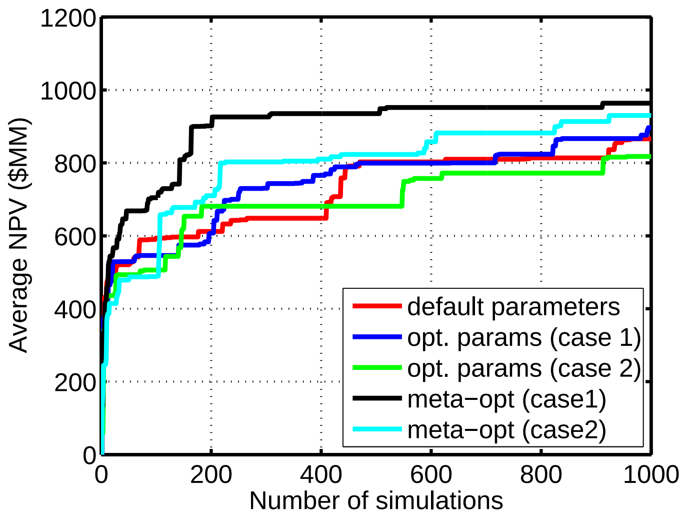


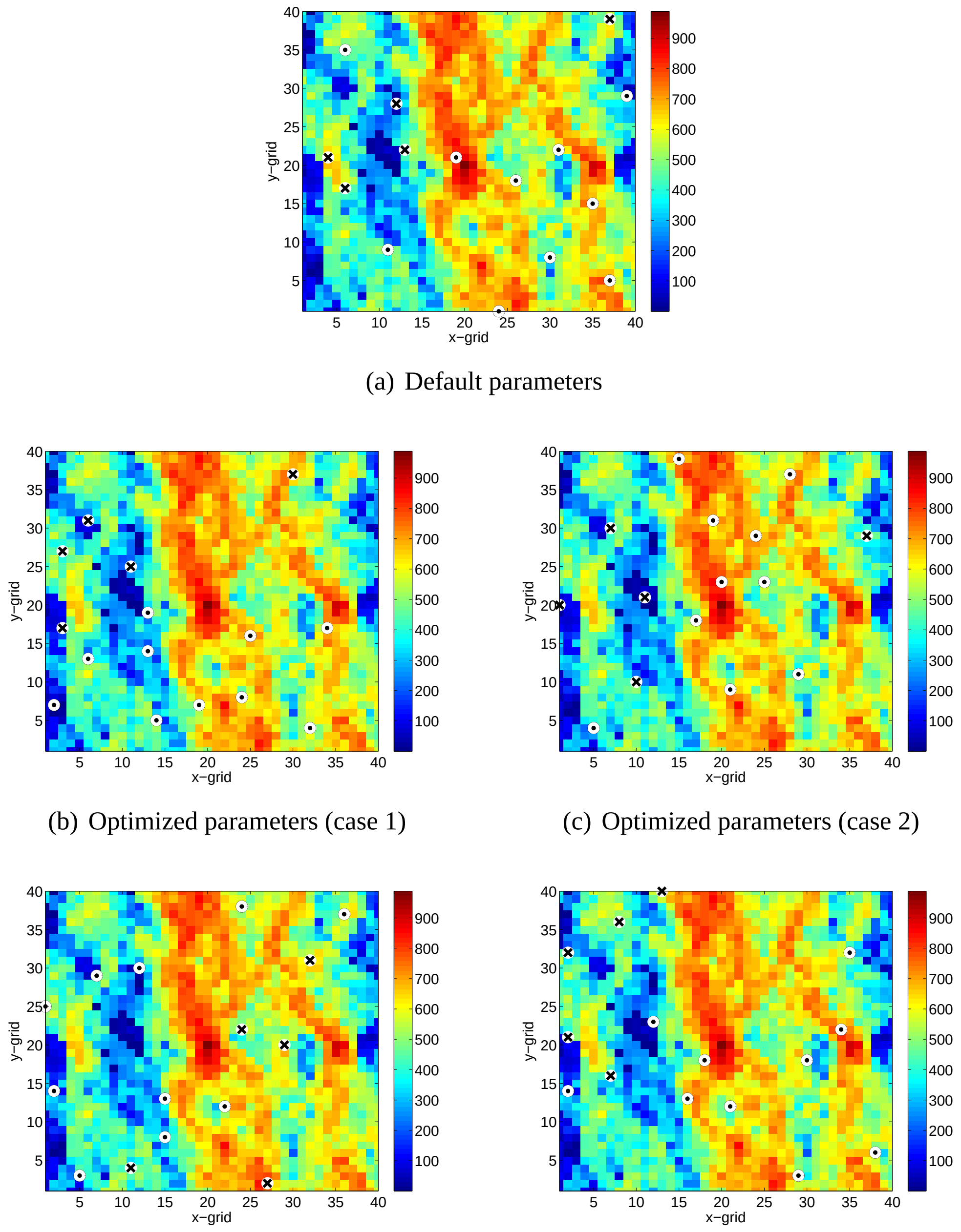
Figure 4.12: Average NPV versus number of simulations for the optimizations using default parameters, optimized parameters and metaoptimization procedure (Example 1).

Table 4.10: Optimization results (NPV) for Example 1. All NPV values have units of $MM

Run	Default	Optimized parameters		Metaoptimization	
		case 1	case 2	case 1	case 2
1	780	961	860	970	938
2	973	791	775	1028	893
3	877	809	813	1032	981
4	961	1006	813	875	891
5	738	917	827	914	948
Average	**866**	**897**	**818**	**964**	**930**

Table 4.11: Metaoptimization results for Example 1

Run	case 1				case 2				
	Avg. Err ($MM)	ω	c_1	c_2	Avg. Err ($MM)	ω	c_1	c_2	n_{type}
1	970	0.918	1.818	1.467	938	0.790	1.468	0.237	3
2	1028	0.750	1.614	1.781	893	0.354	0.283	1.173	4
3	1032	0.945	0.799	1.452	981	0.222	1.536	0.616	2
4	875	0.281	1.297	1.919	891	0.933	0.909	0.769	1
5	914	0.433	1.336	0.754	948	0.356	0.241	0.827	2
Average	**964**				**930**				



(a) Default parameters

(b) Optimized parameters (case 1)

(c) Optimized parameters (case 2)

(d) Metaoptimization (case 1)

(e) Metaoptimization (case 2)

Figure 4.13: Well locations from the best solution in the optimizations using default parameters, optimized parameters and metaoptimization procedure (Example 1). The circles with black dots correspond to producers while the circles with crosses correspond to injectors.

4.4.2 Example 2: WPD optimizations in a 2D reservoir

In this example, we applied the WPD procedure to maximize NPV in a synthetic, heterogeneous two-dimensional model. The reservoir consists of 63×63 grid blocks, each of dimensions of 200 ft$\times$200 ft$\times$50 ft. The permeability field is shown in Fig. 4.14. The initial reservoir pressure is 5000 psi and producers and injectors operate under BHP constraints of 1000 psi and 5200 psi respectively. The other reservoir and fluid properties are the same as those in the example described in Section 3.3.3.

We performed optimizations using the default and optimized parameters. For these runs we used a swarm size of 40 and performed 600 simulations. In the metaoptimization runs, for the superswarm, we set the swarm size to 10 and number of iterations to 5. For the subswarm, we used a swarm size of 20 and and 10 iterations.

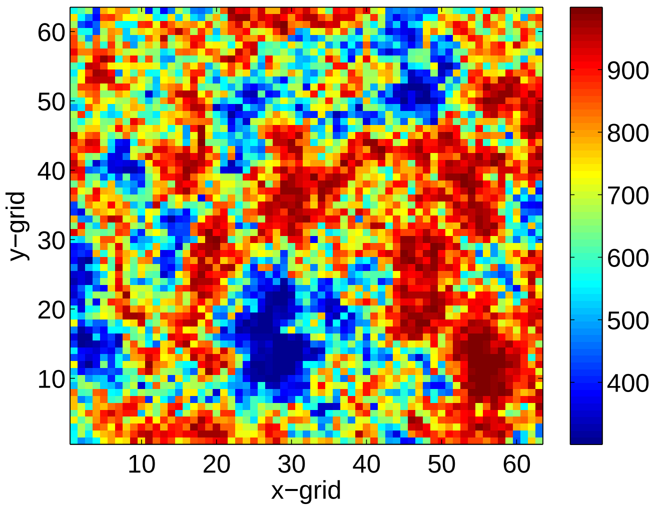


Figure 4.14: Permeability field for Example 2.

Figure 4.15 shows the average NPV of the best solutions versus number of simulations for all the optimization cases. The metaoptimization procedure provides better results than the optimizations using default and optimized parameters. In this case, the metaoptimization procedure again showed a rapid increase in NPV during the early iterations. The results for the optimizations using default and optimized parameters were similar. Again, the case 1 parameters provided slightly better results on average than the default parameters while the case 2 parameters gave lower NPVs. Table 4.12 shows the NPV results for the different optimization runs and Table 4.13 presents the optimized parameters. Figure 4.16 shows the well locations from the best (overall) solution (run 5, case 1) of the metaoptimization runs. The basic well pattern is an inverted six-spot with 6 producers and 18 injectors.

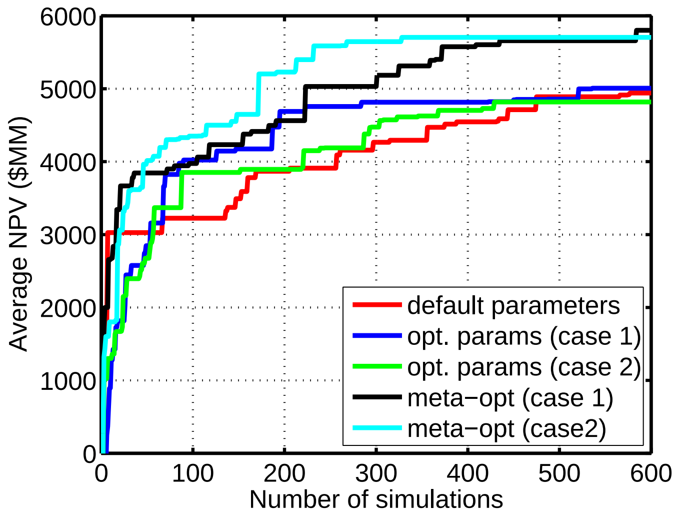


Figure 4.15: Average NPV versus number of simulations for the optimizations using default parameters, optimized parameters and metaoptimization procedure (Example 2).

Table 4.12: Optimization results (NPV) for Example 2. All NPV values have units of $MM

Run	Default	Optimized parameters		Metaoptimization	
		case 1	case 2	case 1	case 2
1	5220	4962	5213	5536	5070
2	3735	5132	4911	5209	6161
3	5837	5111	4263	5910	5603
4	5110	5635	4684	5983	5649
5	4806	4191	5018	6371	6064
Average	**4942**	**5006**	**4818**	**5802**	**5709**

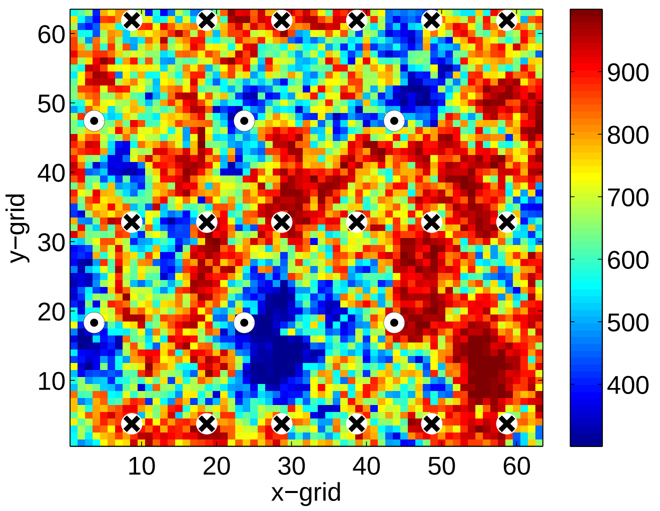


Figure 4.16: Well locations from the best metaoptimization run (Example 2). The circles with black dots correspond to producers while the circles with crosses correspond to injectors.

Table 4.13: Metaoptimization results for Example 2

Run	case 1				case 2				
	Avg. Err ($MM)	ω	c_1	c_2	Avg. Err ($MM)	ω	c_1	c_2	n_{type}
1	5536	0.462	1.058	1.339	5070	0.469	0.531	1.654	3
2	5209	0.967	1.135	1.177	6161	0.501	0.795	1.751	2
3	5910	0.902	0.979	1.451	5603	0.267	0.411	1.520	2
4	5983	0.589	0.876	1.171	5649	0.639	1.214	1.120	2
5	6371	0.543	1.336	1.440	6064	0.810	0.402	0.628	3
Average	**5802**				**5709**				

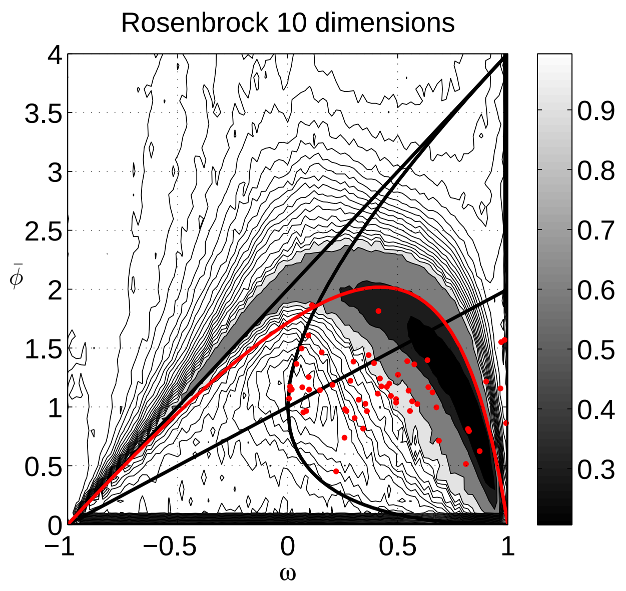
4.5 Further Assessment of the Metaoptimization Results

Here, we compared the values of the optimized PSO parameters (ω, c_1, and c_2) obtained from the metaoptimization runs to those obtained by sampling ω, c_1, and c_2 exhaustively for the Rosenbrock and Griewank mathematical functions. Figure 4.17 shows plots of $\bar{\phi} = 0.5(c_1 + c_2)$ versus ω for both the Rosenbrock and Griewank functions with the contours showing the average objective function values over several runs (the two exhaustive plots are taken from [66]). In Figure 4.17, combinations of $\omega,\ c_1$ and c_2 that result in low objective function values are preferred. We have superimposed the $\omega,\ c_1$ and c_2 values (indicated by the red dots) obtained from all the metaoptimization runs in this chapter. In both plots, most of the values of $\omega,\ c_1$, and c_2 from the metaoptimization are located in the regions with low objective function values. The lines on the plot indicate different regions where a PSO particle trajectory may be convergent or divergent. The interested reader is referred to [3, 49, 50, 66] for more details regarding these regions. In general, the values of $\omega,\ c_1$ and c_2 from the metaoptimization runs agree well with the conditions required for stable and convergent PSO particle trajectories described in [3, 66].

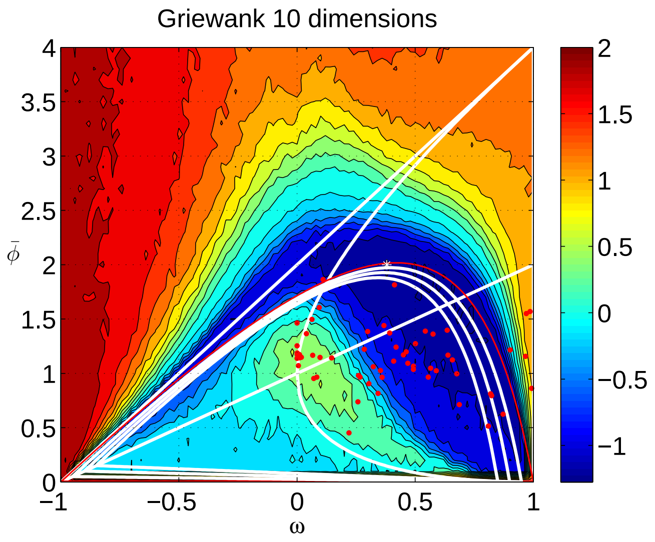
4.6 Summary

In this chapter, we discussed the implementation of the metaoptimization procedure to optimize PSO parameters during optimization. Metaoptimization involves the use of two PSO algorithms, with the first one (superswarm PSO) optimizing the PSO parameters used in the second one (subswarm PSO). The well placement problems were solved in the subswarm PSO using the parameters from the associated superswarm particle. The metaoptimization procedure was applied for well placement optimization using four benchmark problems. We performed optimizations first using each test problem separately, and then using all four problems together. The latter procedure enabled us to determine appropriate parameters for a suite of optimization problems.

We applied the optimized parameters (from the metaoptimizations with the benchmark problems) to realistic and much larger well placement optimizations. Two well placement



(a) Rosenbrock



(b) Griewank

Figure 4.17: Comparison of the PSO parameters from the metaoptimization runs to those obtained by sampling ω, c_1 and c_2 exhaustively for the Rosenbrock and Griewank functions in 10 dimensions. The background plots of $\bar{\phi} = 0.5(c_1 + c_2)$ versus ω were obtained from [66]. The red dots represent the results from all metaoptimization runs.

problems were considered including the optimization of 15 vertical wells and WPD optimizations in two-dimensional reservoir models. The use of optimized parameters from the benchmark problems did not result in optimized NPVs that were significantly larger than those determined using default parameters for the two problems considered.

We also applied the metaoptimization procedure directly to the two optimization problems. We compared metaoptimization results to those from optimizations using the default and optimized parameters for the same number of simulations. For both optimization problems, the best results were obtained using metaoptimization, i.e., when the parameters were directly optimized for the target problems. In addition, metaoptimization showed much faster increases in average NPV in early iterations for the two problems considered. Thus the results in this chapter suggest that metaoptimization should be considered for practical well placement optimization problems.

Chapter 5

Summary and Future Work

In this research, we investigated the use of the particle swarm optimization (PSO) algorithm to optimize the type and location of new wells. We addressed the problem of optimizing well locations in large-scale field development involving many wells. We also addressed issues related to the determination of optimum PSO parameters for well placement optimization. The findings in this dissertation are also relevant for other optimization problems, e.g., production optimization, and optimizing injection and extraction wells for contaminated groundwater applications.

5.1 Summary and Conclusions

5.1.1 Particle Swarm Optimization

- The particle swarm optimization (PSO) algorithm was implemented and applied to the optimization of well locations and type. The algorithm is stochastic and population-based. The individual solutions, called particles, interact with each other and exchange information regarding the search space. Particles interact with other particles in their neighborhood. We implemented different particle neighborhood topologies, which can affect algorithm performance.

- The PSO algorithm was applied to several well placement optimization problems of varying complexity. We considered problems with different numbers of wells, different types of wells (vertical, deviated, multilateral), and different sizes of the search space. Multiple geological realizations were used in some cases. We compared the optimization results to those obtained using a binary GA (bGA). In all examples considered, we demonstrated that the PSO algorithm provided comparable or better results on average than the bGA. For a case in which the global optimum was known (through exhaustive sampling), PSO was shown to achieve the global minimum with fewer function evaluations than bGA. For this case, the performance of the PSO and GA algorithms improved when the swarm/population size and/or the number of iterations were increased. Our findings regarding the performance of PSO are consistent with those from a related application involving the optimization of extraction well locations in contaminated groundwater applications [63].

- The PSO algorithm was used also for other applications. It was used as the core optimization algorithm in the well pattern optimization (WPO) procedure and was applied for optimizing PSO parameters in the metaoptimization procedure.

5.1.2 Well Pattern Optimization

- A new procedure for optimizing well placement in large-scale field developments involving many wells was developed. The new algorithm, called well pattern optimization (WPO), consists of a well pattern description (WPD) incorporated into a core optimization algorithm. In the well pattern descriptions, each solution consists of a representation of a particular well pattern along with pattern operators that alter the size, shape, and orientation of the pattern. Many different well patterns can be considered within WPD. It is the parameters associated with the pattern descriptions and operators that are determined during the optimizations. The encoded well patterns are repeated across the field, which enables the optimum number of wells to be determined as part of the solution. A desirable feature of WPD is that the computational complexity of the optimization is essentially independent of the number of wells considered.

- A well-by-well perturbation (WWP) procedure was also developed. WWP, which can be applied as an optional second phase of the optimization, entails a local perturbation of the well locations obtained from the WPD optimization. For the underlying (core) optimization algorithm, we used particle swarm optimization (PSO).

- The WPO procedure was applied to four example cases. Several variants of WPO were considered including the use of one versus four operators for each potential solution and the use of WWP following optimization using WPD. The overall optimization procedure was shown to result in significant increases in the objective function, particularly at early iterations, in all cases. In one example the WPO results for net present value (NPV) were compared to those for standard well patterns of various sizes. The NPVs using WPO were seen to be significantly larger than those for standard well patterns, highlighting the potential benefit of the algorithm for identifying promising development scenarios. Significant improvement in NPV was obtained by performing WWP optimizations on the best solution obtained using WPD. For the two examples in which WWP was applied, average improvements in NPV of 22% and 34% over the best WPD solutions were achieved. We also compared WPD results to those obtained from optimizations using concatenated well parameters and found the WPO procedure to provide better solutions.

5.1.3 Metaoptimization for Parameter Determination

- A procedure for determining optimum PSO parameters was implemented and tested. This procedure, called metaoptimization, optimizes PSO parameters during optimization. Metaoptimization requires two PSO algorithms, where the first algorithm optimizes PSO parameters, and the second algorithm optimizes a given optimization problem or problems using the PSO parameters from the first algorithm.

- We applied the metaoptimization procedure to determine optimum PSO parameters using four benchmark well placement optimization problems. In the metaoptimizations, we considered two cases. In case 1, we optimized PSO parameters $\omega,\ c_1,\ c_2$, while in case 2, we optimized the neighborhood topology type in addition to $\omega,\ c_1,\ c_2$. We showed that metaoptimization provided results better than those for PSO with unoptimized parameters. Next, we applied the metaoptimization to all four benchmark problems in order to determine PSO parameters that are optimal for multiple problems.

- We applied the optimized PSO parameters and metaoptimization to two optimization problems including optimizing the location of 15 vertical wells (consisting of ten producers and five injectors) and WPD optimizations in heterogeneous two-dimensional reservoirs. For these examples, metaoptimization was shown to provide the best results.

5.2 Recommendations for Future Work

- In the PSO optimizations in Chapter 2, we represent solutions using well-by-well concatenation. For these optimizations, the PSO algorithm did not include any constraint handling. If well-by-well parameter concatenation is to be used, it will be necessary to incorporate constraint handling methods in order to obtain practically acceptable well configurations. This can be achieved by several methods. One approach is to use penalty function methods and penalize all infeasible solutions. We used this method for an example problem in Chapter 4. Other approaches include feasibility preserving and repair methods [3]. In feasibility preserving methods, the particle positions are computed so that constraints are not violated. In repair methods, special operators are used to repair an infeasible particle or to adjust it so it satisfies all constraints. A similar technique was implemented for GA in [29] to handle general well placement optimization constraints. Other constraint handling techniques for PSO are described in [3].

- Potentially the performance of the PSO algorithm could be improved further by incorporating methods to increase or preserve the diversity of the particles. As the PSO particles converge, they may exhibit a tendency to cluster around the best solution, especially when the star neighborhood topology is used. Procedures for improving the diversity of particles should be explored. For example, if the objective function does not improve over a prescribed number of iterations, a fraction of the particles could be reinitialized around their local best positions [3].

- The PSO performance could also be improved by balancing exploration and exploitation within the optimization. One way to achieve this is to use a generalized PSO where Δt (in the velocity and position update equations) can take values other than unity [50, 49].

- It may be of interest to investigate specialized PSO particle neighborhood topologies. Also, other variants of the two-phase optimization strategy (WPD followed by WWP) should be considered. In addition, the number of patterns considered could be increased to include other well patterns, e.g., the 13-spot well pattern. Such additions do not lead to an increase in the number of variables required to represent the patterns in WPD.

- The WPO procedure should be applied to larger models (in areal extent) than those considered here. It will also be of interest to test the WPO procedure for practical problems. For such cases, it may be useful to extend the WWP procedure to optimize completion interval, to allow for the elimination of particular wells, etc.

- The efficiency of the PSO and WPO algorithms may be improved through the use of surrogate (proxy) simulation models, e.g., kriging, statistical proxies, and neural networks. This would act to reduce the number of time consuming simulations required. Use of a hybrid optimization approach involving the combination of PSO with a local optimizer may also prove effective.

- The metaoptimization procedure was demonstrated to be effective at optimizing PSO parameters and providing better solutions, on average, relative to those achieved using untuned/unoptimized PSO parameters. However, the procedure is computationally intensive because of the many function evaluations required. Surrogate models should be incorporated into the metaoptimization procedure to reduce the computational demands and to enable this approach to be used for practical applications.

- The combined optimization of well placement and well control should be considered. This will be very demanding computationally, so efficient optimization strategies will need to be devised.

Nomenclature

Roman Symbols

$a,\ b$	well spacing parameters
C^{capex}	capital expenditure, \$
C^{drill}	drilling cost within the reservoir, \$/ft
C^{junc}	junction cost of a lateral, \$
C_w^{top}	cost to drill the main bore to top of reservoir, \$
$c_1,\ c_2$	weight of cognitive and social components in PSO velocity equation
CF	cash flow, \$
c	compressibility, psi^{-1}
d	number of optimization variables
E	operating expense, \$
F	objective function of superswarm particle
H_ξ, H_η	axis shearing factors
I^{wp}	index of well pattern
k	index of iteration
l	lower bound of a variable
L^{lat}	length of a lateral, ft
L^{main}	length of the main bore, ft
$\mathbf{M}$	transformation matrix
$\mathbf{M}_\theta,\ \mathbf{M}_{sc},\ \mathbf{M}_{sh}$	rotation, scale, and shear transformation matrices
$\langle$NPV$\rangle$	expected net present value
$\mathcal{N}_o$	number of pattern operators

n	index of a well
N^{lat}	number of laterals in well
N^{wells}	number of wells
N_I	number of particles informed
N_p	number of different well patterns
N_s	swarm size
N_{wp}	number of wells in a well pattern
n_{type}	neighborhood topology type
$\mathcal{O}$	pattern operator
$\mathcal{P}$	pattern representation
P	number of optimization problems
p	index of optimization problem or price per unit volume, \$/STB, \$/SCF
Q	total volume of fluid produced
$\mathbf{D}_1, \mathbf{D}_2$	diagonal matrices of random numbers between 0 and 1
R	number of subswarm optimizations or revenue, \$
r	discount rate
S	operator sequence
S_ξ, S_η	axis scaling factors
$S_{gi},\ S_{oi},\ S_{wi}$	initial gas, oil, and water saturations
Δt	time increment
T	total production time or number of iterations in subswarm
t	production time or index of iteration
u	upper bound of a variable
$\mathbf{v}$	PSO particle velocity
$\mathbf{W}$	matrix of well locations in a well pattern
$\mathbf{X}$	superswarm PSO particle position
$\mathbf{x}$	(subswarm) PSO particle position

Greek Symbols

μ	viscosity, cp
ω	inertia weight

θ	rotation angle
$\Delta\xi,\ \Delta\eta$	spatial perturbations of a well location
$\xi,\ \eta$	areal location of a well
$\xi^0,\ \eta^0$	center of a well pattern

Superscripts

0	PSO parameters from superswarm particle
c	cognitive
g	global best
g, o, w	gas, oil, and water phases
$nbest$	neighborhood best
$pbest$	previous best
ref	reference
s	social
wp	well pattern

Subscripts

g, o, w	gas, oil, and water phases
i	index of PSO particle or injected water
j	index of pattern operator or index of optimization variable
l	index of lateral
r	rock
rot	rotation
t	production period
w	index of well

Abbreviations

BHP	bottomhole pressure, psi
bGA	binary genetic algorithm
GA	genetic algorithm
gbest	global best
GPRS	general purpose research simulator

lbest	local best
MM, $MM	million, million dollars
NPV	net present value
PSO	particle swarm optimization
SimA	simulated annealing
SPSA	simultaneous perturbation stochastic approximation
WPD	well pattern description
WPO	well pattern optimization
WWP	well-by-well perturbation

References

[1] Y. Collette and P. Siarry. *Multiobjective Optimization: Principles and Case Studies*. Springer-Verlag, Berlin, Germany, 2003.

[2] M. Clerc. *Particle Swarm Optimization*. iSTE, London, England, 2006.

[3] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, West Sussex, England, 2005.

[4] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[5] B. L. Beckner and X. Song. Field development planning using simulated annealing - optimal economic well scheduling and placement. Paper SPE 30650 presented at the SPE Annual Technical Conference and Exhibition, Dallas, Texas, U.S.A., 22-25 October, 1995.

[6] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer-Verlag, Berlin, Germany, 2004.

[7] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, Inc., New Jersey, U.S.A., 2004.

[8] W. Bangerth, H. Klie, M. F. Wheeler, P. L. Stoffa, and M. K. Sen. On optimization algorithms for the reservoir oil well placement problem. *Computational Geosciences*, 10:303–319, 2006.

[9] K. P. Norrena and C. V. Deutsch. Automatic determination of well placement subject to geostatistical and economic constraints. Paper SPE 78996 presented at the SPE International Thermal Operations and Heavy Oil Symposium and International Horizontal Well Technology Conference, Calgary, Alberta, Canada, 4-7 November, 2002.

[10] K. L. Franstrom and M. L. Litvak. Automatic simulation algorithm for appraisal of future infill development potential of Prudhoe Bay. Paper SPE 59374 presented at the SPE/DOE Improved Oil Recovery Symposium, Tulsa, Oklahoma, U.S.A., 3-5 April, 2000.

[11] M. Litvak, B. Gane, G. Williams, M. Mansfield, P. Angert, C. Macdonald, L. McMurray, R. Skinner, and G.J. Walker. Field development optimization technology. Paper SPE 106426 presented at the SPE Reservoir Simulation Symposium, Houston, Texas, U.S.A., 26-28 February, 2007.

[12] G. J. J. Williams, M. Mansfield, D. G. Macdonald, and M. D. Bush. Top-down reservoir modeling. Paper SPE 89974 presented at the SPE Annual Technical Conference and Exhibition, Houston, Texas, U.S.A., 26-29 September, 2004.

[13] V. Artus, L. J. Durlofsky, J. E. Onwunalu, and K. Aziz. Optimization of nonconventional wells under uncertainty using statistical proxies. *Computational Geosciences*, 10(4):389–404, 2006.

[14] A. C. Bittencourt and R. N. Horne. Reservoir development and design optimization. Paper SPE 38895 presented at the SPE Annual Technical Conference and Exhibition, San Antonio, Texas, U.S.A., 5-8 October, 1997.

[15] M. Morravvej Farshi. Master's thesis. Improving genetic algorithms for optimum well placement, M.S. thesis, Department of Energy Resources Engineering, Stanford University, 2008.

[16] B. Guyaguler and R. N. Horne. Uncertainty assessment of well placement optimization. Paper SPE 71625 presented at the SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, U.S.A., 30 September - 3 October, 2001.

[17] B. Guyaguler, R. N. Horne, L. Rogers, and J. J. Rosenzweig. Optimization of well placement in a Gulf of Mexico waterflooding project. *Reservoir Evaluation and Engineering*, 5(3):229–236, 2002.

[18] J. E. Onwunalu. Master's thesis. Optimization of nonconventional well placement using genetic algorithms and statistical proxy, M.S. thesis, Department of Energy Resources Engineering, Stanford University, 2006.

[19] Y. J. Tupac, L. Faletti, M. A. C. Pacheco, and M. M. B. R. Vellasco. Evolutionary optimization of oil field development. Paper SPE 107552 presented at the SPE Digital Energy Conference and Exhibition, Houston, Texas, U.S.A., 11-12 April, 2007.

[20] B. Yeten, L. J. Durlofsky, and K. Aziz. Optimization of nonconventional well type, location and trajectory. *SPE Journal*, 8(3):200–210, 2003.

[21] U. Ozdogan and R. N. Horne. Optimization of well placement under time-dependent uncertainty. *Reservoir Evaluation and Engineering*, 9(2):135–145, 2006.

[22] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, U.S.A., 1989.

[23] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, Germany, 1996.

[24] S. N. Sivananadam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer-Verlag, Berlin, Germany, 2008.

[25] A. Y. Abukhamsin. Optimization of well design and location in a real field, M.S. thesis, Department of Energy Resources Engineering, Stanford University, 2009.

[26] O. Badru and C. S. Kabir. Well placement optimization in field development. Paper SPE 84191 presented at the SPE Annual Technical Conference and Exhibition,

Denver, Colorado, U.S.A., 5-6 October, 2003.

[27] G. Montes, P. Bartolome, and A. L. Udias. The use of genetic algorithms in well placement optimization. Paper SPE 69439 presented at the SPE Latin American and Carribean Petroleum Engineering Conference held in Buenos Aires, Argentina, 25-28 March, 2001.

[28] B. Yeten. *Optimum deployment of nonconventional wells*. PhD thesis, Stanford University, 2003.

[29] A. A. Emerick, E. Silva, B. Messer, L. F. Almeida, D. Szwarcman, M. A. C. Pacheco, and M. M. B. R. Vellasco. Well placement optimization using a genetic algorithm with nonlinear constraints. Paper SPE 118808 presented at the Reservoir Simulation Symposium, The Woodlands, Texas, U.S.A., 2-4 February, 2009.

[30] D. Y. Ding. Optimization of well placement using evolutionary algorithms. Paper SPE 113525 presented at the SPE Europec/EAGE Annual Conference and Exhibition, Rome, Italy, 9-12 June, 2008.

[31] A. Larionov, A. Nifantov, V. Itkin, and V. Alexandrov. Methodology of optimal well pattern, location and paths in productive formations during oil and gas field development planning. Paper SPE 104326 presented at the SPE Russian Oil and Gas Technical Conference and Exhibition, Moscow, Russia, 3-6 October, 2006.

[32] U. Ozdogan, A. Sahni, B. Yeten, B. Guyaguler, and W. H. Chen. Efficient assessment and optimization of a deepwater asset development using fixed pattern approach. Paper SPE 95792 presented at the SPE Annual Technical Conference and Exhibition, Dallas, Texas, U.S.A., 9-12 October, 2005.

[33] J. C. Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins Applied Physics Laboratory Technical Digest*, 19(4):482–492, 1998.

[34] R. D. Hazlett and D. K. Babu. Optimal well placement in heterogeneous reservoirs through semianalytic modeling. *SPE Journal*, 10(3):286–296, 2005.

[35] C. Wang, G. Li, and A. C. Reynolds. Optimal well placement for production optimization. Paper SPE 111154 presented at the 2007 SPE Eastern Regional Meeting, Lexington, Kentucky, U.S.A., 11-14 October, 2007.

[36] M. J. Zandvliet, M. Handels, G. M. van Essen, D. R. Brouwer, and J. D. Jansen. Adjoint-based well-placement optimization under production constraints. *SPE Journal*, 13(4):392–399, 2008.

[37] P. Sarma and W. H. Chen. Efficient well placement optimization with gradient-based algorithms and adjoint models. Paper SPE 112257 presented at the SPE Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 25-27 February, 2008.

[38] S. Vlemmix, G. J. P. Joosten, D. R. Brouwer, and J. D. Jansen. Adjoint-based well trajectory optimization in a thin oil rim. Paper SPE 121891 presented at the SPE EUROPEC/EAGE Annual Conference and Exhibition, Amsterdam, The Netherlands, 8-11 June, 2009.

[39] Y. Pan and R. N. Horne. Improved methods for multivariate optimization of field development scheduling and well placement design. Paper SPE 49055 presented at the SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, U.S.A., 27-30 September, 1998.

[40] A. Centilmen, T. Ertekin, and A. S. Grader. Applications of neural networks in multiwell field development. Paper SPE 56433 presented at the SPE Annual Technical Conference and Exhibition, Houston, Texas, U.S.A., 3-6 October, 1999.

[41] J. E. Onwunalu, M. L. Litvak, L. J. Durlofsky, and K. Aziz. Application of statistical proxies to speed up field development optimization procedures. Paper SPE 117323 presented at the Abu Dhabi International Petroleum Exhibition and Conference, Abu Dhabi, U.A.E., 3-6 November, 2008.

[42] F. Zarei, A. Daliri, and N. Alizadeh. The use of neuro-fuzzy proxy in well placement optimization. Paper SPE 112214 presented at the SPE Intelligent Energy Conference

and Exhibition, Amsterdam, The Netherlands, 25-27 February, 2008.

[43] N. Liu and Y. Jalali. Closing the loop between reservoir modeling and well placement and positioning. Paper SPE 98198 presented at the SPE Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 11-13 April, 2006.

[44] R. Volz, K. Burn, M. Litvak, S. Thakur, and S. Skvortsov. Field development optimisation of Siberian giant oil field under uncertainties. Paper SPE 116831 presented at the SPE Russian Oil and Gas Technical Conference and Exhibition, Moscow, Russia, 28-30 October, 2008.

[45] M. L. Litvak and P. F. Angert. Field development optimization applied to giant oil fields. Paper SPE 118840 presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, U.S.A., 2-4 February, 2009.

[46] M. L. Litvak, M. L. Gane, and L. McMurray. Field development optimization in a giant oil field in Azerbaijan and a mature oil field in the North Sea. Paper OTC 18526 presented at the Offshore Technology Conference, Houston, Texas, U.S.A., 30 April-3 May, 2007.

[47] J. Kennedy and R. C. Eberhardt. Particle swarm optimization. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1942-1947, IEEE, December, 1995.

[48] R. C. Eberhardt and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, pages 39–43, 1995.

[49] J. L. F. Martinez and E. G. Gonzalo. The generalized PSO: a new door to PSO evolution. *Artificial Evolution and Applications*, 2008:1–15, 2008.

[50] J. L. F. Martinez and E. G. Gonzalo. The PSO family: deduction, stochastic analysis and comparison. *Swarm Intelligence*, 3(4):245–273, 2009.

[51] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in

genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[52] R. L. Haupt. Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors. In *Antennas and Propagation Society International Symposium*, volume 2, pages 1034–1037. IEEE, 2000.

[53] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.

[54] J. Kennedy and R. Mendes. Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms. In *Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications*, pages 45–50. IEEE Press, June 2003.

[55] V. Miranda, H. Keko, and A. J. Duque. Stochastic star communication topology in evolutionary particle swarms (EPSO). *International Journal of Computational Intelligence Research*, 4(2):105–116, 2008.

[56] D. Braendler and T. Hendtlass. The suitability of particle swarm optimisation for training neural hardware. In *International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE*, pages 190–199. Springer, 2002.

[57] R. Chakrabarti, P. K. Chattopadhyay, M. Basu, and C. K. Pangrahi. Particle swarm optimization technique for dynamic economic dispatch. *IE(I) Journal-EL*, 87:48–54, 2006.

[58] B. Brandstatter and U. Baumgartner. Particle swarm optimization–mass-spring system analogon. *IEEE Transactions on Magnetics*, 38(2):997–1000, 2002.

[59] D. N. Kumar and M. J. Reddy. Multipurpose reservoir operations using particle swarm optimization. *Water Resources Planning and Management*, 133(3):192–201, 2007.

[60] A. R. M. Rao and G. Anandakumar. Optimal placement of sensors for structural system identification and health monitoring using a hybrid swarm intelligence technique.

Smart Materials and Structures, 16:2658–2672, 2007.

[61] J. L. F. Martinez and E. G. Gonzalo. Design of a simple and powerful particle swarm optimizer. Proceedings of the International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE2009, Gijon, Span, 30 June-3 July, 2009.

[62] O. Uysal and S. Bulkan. Comparison of genetic algorithm and particle swarm optimization for bicriteria permutation flowshop scheduling problem. *International Journal of Computational Intelligence Research*, 4(2):159–175, 2008.

[63] L. S. Matott, A. J. Rabideau, and J. R. Craig. Pump-and-treat optimization using analytic element method flow models. *Advances in Water Resources*, 29:760–775, 2006.

[64] M. B. Abdelhalim and S. E. D. Habib. Particle swarm optimization for HW/SW partitioning. In Particle Swarm Optimization, edited by A. Lazinica, pages 49-76, 2009.

[65] Y. Shi and R. C. Eberhardt. A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69-73, IEEE Press, May, 1998.

[66] J. L. F. Martinez, E. G. Gonzalo, and J. P. F. Alvarez. Theoretical analysis of particle swarm trajectories through a mechanical analogy. *International Journal of Computer Intelligence Research*, 2(2-4):93–104, 2006.

[67] M. Clerc and J. Kennedy. The particle swarm – explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computations*, 6(1):58–73, 2002.

[68] M. Clerc. Stagnation analysis in particle swarm optimization or what happens when nothing happens. 2006. Technical Report CSM-460, Department of Computer Science, University of Essex. Edited by Riccardo Poli.

[69] M. Meissner, M. Schmuker, and G. Schneider. Optimized particle swarm optimization (OPSO) and its application to artificial neural network training. *BMC Bioinformatics*, 7(125):1–11, 2006.

[70] H. Zhang and M. Ishikawa. Evolutionary canonical particle swarm optimizer - a proposal of meta-optimization in model selection. In V. Kurková, R. Neruda, and J. Koutnik, editors, *Artificial Neural Networks - ICANN 2008, Part I, LNCS 5163*, volume 5163/2008, pages 472–481. Springer-Verlag, 2008.

[71] H. Zhang and M. Ishikawa. Evolutionary particle swarm optimization: a metaoptimization method with GA for estimating optimal PSO models. In O. Castillo, L. Xu, and S. Ao, editors, *Trends in Intelligent Systems and Computer Engineering*, volume 6, pages 75–90. Springer US, 2008.

[72] J. E. Onwunalu and L. J. Durlofsky. Application of a particle swarm optimization algorithm for determining optimum well location and type. *Computational Geosciences*, 14(1):183–198, 2010.

[73] J. E. Onwunalu and L. J. Durlofsky. Development and application of a new well pattern optimization algorithm for optimizing large-scale field development. 2009. Paper SPE 124364 presented at the SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, USA 4-7 October.

[74] S. Helwig and R. Wanka. Theoretical analysis of initial particle swarm behavior. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN08)*, pages 889–898. Dortmund, Germany, Springer, 13-17 September, 2008.

[75] W. J. Zhang, X. F. Xie, and D. C. Bi. Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 2307–2311,. IEEE, 2004.

[76] A. Carlisle and G. Dozier. An off-the-shelf PSO. In *Proceedings of Workshop on Particle Swarm Optimization*. Indianapolis, IN, U.S.A., 6-7 April, 2001.

[77] H. Cao. *Development of techniques for general purpose research simulator*. PhD thesis, Stanford University, 2002.

[78] Y. Jiang. *Techniques for modeling complex reservoirs and advanced wells*. PhD thesis, Stanford University, 2007.

[79] Schlumberger Eclipse. Schedule User Guide 2004A, Chapter 6: Technical description, 2004.

[80] J. T. Alander. On optimal population size of genetic algorithms. In *Proceedings of CompEuro '92 Computer Systems and Software Engineering*, pages 65–70, 1992.

[81] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, West Sussex, England, 2001.

[82] F. R. Craig. *The Reservoir Engineering Aspects of Waterflooding*. SPE Monograph, Volume 3, 1971.

[83] 3DSL. v2.30 User Manual. Streamsim Technologies Inc., San Francisco, California, U.S.A., 2006.