Evolutionary Algorithms-Based Computational Framework for Solving Inverse Problems

Baha Y. Mirghani
Department of Civil, Construction and Environmental Engineering
North Carolina State University, Raleigh, NC 27695
+1-919-515-4342
bmirgha@ncsu.edu

ABSTRACT
Inverse problems are relatively challenging to solve due to inherent ill-posedness and computational intractability. In this paper we adopt the use of a simulation-optimization approach that couples a numerical simulation model with evolutionary algorithms for solution of the inverse problem. In this approach, the simulation model is solved iteratively during the evolutionary search, which in general can be computationally intensive since several hundreds to thousands of forward model evaluations are typically required for solution. Numerical search methods such as parallel hybrid methods and noisy genetic algorithms are investigated for optimization algorithm improvement. Given the potential computational intractability of such a simulation-optimization approach, grid computing and surrogate models are explored as a means to facilitate computationally tractable solution of such problems. In this paper, the solution of a groundwater inverse problem is explored to test and illustrate the methods. The computational experiments were performed on the National Scientific Foundation's TeraGrid. The results demonstrate the performance of the grid-enabled simulation-optimization approach in terms of solution quality and computational performance. A set of preliminary results from ongoing research is discussed.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**] *Heuristic methods,* D.3.2 [**Programming Languages**], G.4 [**Mathematical Software**]

General Terms
Algorithms, Management, Design

Keywords
Inverse problem, Simulation-Optimization, Algorithms, Grid Computing, Surrogate Modeling

1. INTRODUCTION
System characterization from sparse observational data collected at measurement stations is generally classified as an inverse problem. Solving inverse problems is relatively complex due to the ill-posedness present in the problem [1] [15]. Among several available methods, simulation-optimization (S-O) approach is a technique utilized to solve inverse problems by formulating and solving them as an optimization model. Simulation models, usually referred to as forward models, are a system of partial differential equations (PDEs) that describes the governing processes of a system and defines the relationship between system inputs and outputs. The inverse problem is solved using an S-O approach via search algorithms to identify the best system characteristics that minimize the error between the model predictions and the system observations. While this approach is generic and robust, it is computationally expensive as it requires iterative executions of the forward model.

Many instances of inverse problems are present in real world applications. For example, in groundwater contamination characterization problems, the unknown contamination location and the release profile are estimated based on observation data. Similarly, in drinking water distribution system security management, monitoring data from a contamination sensor network is used to determine the contamination location and characteristics so that appropriate evasive actions could be determined quickly. Another application is in medical imaging where the exact properties of internal organs are assessed via non-invasive screening and diagnosis.

In this study, instances of groundwater containment source identification (CSI) problems are used to test and illustrate the methodologies. Also the problems utilized to test the performance of the computational framework that developed as part of my dissertation research. The CSI is important in environmental forensics and characterization of contamination for the purposes of regulatory enforcement and assessing liability. In this problem context, source locations and contaminant release concentration are unknown model inputs, which are resolved from the spatially and temporally distributed observational data collected at monitoring wells. This paper discusses the solution of the containment source identification problem in two and three dimensional groundwater domains.

The solution complexity of such problems is proportional to the number of system inputs to be determined. An optimization search paired with a detailed numerical simulation model poses

extreme computational burdens since each simulation model execution can be expensive. In this study, evolutionary algorithms-based search procedures are employed to solve inverse problems using an S-O approach. The focus of my dissertation is to investigate novel approaches to address the algorithmic and related computational issues associated with the proposed solution method. This consists of three major components: improving the effectiveness and efficiency of the search algorithms; improving the execution efficiency of the simulation model; and facilitating a distributed computational framework to support the S-O approach.

While evolutionary algorithms (EAs) are good global searchers, they are inefficient, when compared with local search procedures, in refining the solutions beyond a certain degree of convergence. This research investigates new techniques for integrating the good features of the global and local searchers such that the overall search efficiency is improved. In real world application, inverse problems occur under noisy environment. Thus, another methodological investigation of this dissertation research is to develop a new algorithm for search under noisy conditions. In addition, this research investigates techniques to improve the computational efficiency of the simulation model. A fine-grained parallel coding of the simulation model and a fast surrogate model to approximate the simulation model are being explored.

Finally, given the computational resources demands and the inherent parallelism present in the proposed S-O approach, an appropriate parallel/distributed computing framework can help improve the computational efficiency. Recent investments in national high-speed network infrastructure have allowed the aggregation of geographically distributed high-performance computing resources into computational grids. In part, because computational grids promote reliable and economical access to, and sharing of, high-end computing resources, they have emerged as a new paradigm in scientific and engineering computation [6]. Computational grids have the potential to enable the solution of inverse problems that previously would not have been possible. Results based on computational experiments performed on the National Science Foundation's (NSF) TeraGrid demonstrate the efficiency of the grid-enabled S-O approach in terms of accurately solving the unknown system inputs and improving the computational performance.

Section 2 describes the overall problem complexity and the supporting methodology. The new algorithms are presented in Section 3. Methods for efficiency improvements in simulation models are discussed in Section 4. The computational framework and paradigms are explained in Section 5. Section 6 describes an instance of the groundwater contaminant source characterization problems, followed by preliminary results in Section 7. Section 8 provides some final remarks along with a discussion on on-going and future work.

2. PROBLEM COMPLEXITY AND METHODOLOGY

Generally speaking, inverse problems could be described as problems where the answer is known, but not the conditions that led to it. In mathematical modeling, finding model input parameter values given output data can be categorized as a problem of time inversion. This means that we have to solve the governing equations backward in time through "inverse modeling" [1]. The process of determining system characteristics from observation data is known as inverse problems. Inverse problems are generally difficult to solve due to ill-posedness. A problem is categorized as ill-posed if: (1) the solution does not exist; (2) the solution is nonunique; and (3) the solution is unstable [1][15]. Solving inverse problem is based on observation data, thus the solution have discontinuous reliance on data and are sensitive to errors in the data. In real world applications, errors in the observation could result from data measurement error or model parameter uncertainty. Solving inverse problems in a noisy environment adds another level of complexity to the problems.

Research over the past three decades has presented several solution methods for inverse problems, including simulation-optimization techniques, probabilistic procedures, analytical methods, and direct inversion approaches [8]. This dissertation research explores solution using the simulation-optimization approach. This is a general term used to describe a family of optimization techniques that utilizes simulation models for the evaluation of objective and constraint functions. In this approach, the simulation model is coupled loosely/tightly with optimization techniques to determine the model inputs that best approximate the observed data. This process is applied iteratively until some stopping criteria are met. Several different search procedures have been explored, however, and the emphasis of this study is on EA-based heuristic search approaches.

An S-O approach to solve an inverse problem is several orders of magnitude more computationally challenging than the solution of the corresponding forward model, since several hundreds to thousands of forward model evaluations, usually requiring solution to a system of PDFs, are typically required. Given the potential computational intractability of such an S-O approach, improving efficiency for both simulation model and optimization method is required.

Recent research had shown that inverse problems may have several local minima and a highly discontinuous decision space. Evolutionary algorithms show a significant efficiency in getting near global minimum; however, they may require fine-tuning near global minimum for better results. Hybrid methods show considerable success in this aspect, where EAs may be efficient in getting near global optimum and gradient-based local optimization approaches are efficient for fine-tuning near that global optimum [10]. Since the solution of inverse problems is highly dependent on the errors in the measurements, the search methods must consider noise in the system. Noisy GAs show a considerable success in addressing more realistic inverse problems [11]. Therefore investigating embedded hybrid and noisy GA methods is becoming essential to enable an S-O approach.

High performance computing technologies can help expedite the execution time by harnessing the fine and coarse gained parallelism exhibited by the simulation model. In addition to simulation model parallelism, surrogate models are explored to improve the simulation model efficiency by reducing the solution time. Ultimately a computational grid-based S-O framework with multiple levels of parallelism is used to ease the solution of inverse problem. We would like to highlight that some components of this framework is currently under development.

3. GA-BASED OPTIMIZATION METHODS

In this section, the development of embedded hybrid and noisy genetic algorithms methods are described. While there are several hybridization techniques, the emphasis of my dissertation research is on algorithms that directly embed the local search steps into any standard genetic algorithm such that the inherent parallelism of GAs is maintained [16]. Similarly, my research is also investigating a new approach for a noisy GA, which in conjunction with the embedded hybrid procedure or by itself, is applicable to solving inverse problems under conditions of uncertainty.

3.1 Embedded Hybrid Methods

Previous studies had shown that sequential hybrid methods are successful in solving inverse problems [10][16]. While they are effective, these sequential methods do not necessarily maintain the parallelism to attain maximum computational efficiency. Alternatively, embedded hybrid methods are designed such that the local search steps are integrated into the global search mechanism of a GA. In the following section, two parallel hybrid methods are discussed.

3.1.1 Hooks-Jeeves GA Procedure

Prior research efforts [16] report that the crossover operator could be adjusted to increased local exploitation. In this procedure an alternative crossover operator is developed. This operator is designed based on the Hooks-Jeeves pattern search that explores the immediate neighborhood of a selected solution [5]. The new operator is applied to a subset of the population while the rest of the population undergoes the standard GA crossover operator. The subset size increases dynamically with the number of generations. The main steps of the algorithm are described in the following pseudo code:

```
Hooks-Jeeves GA{
  while {termination criteria is not met
    Evaluate the population ();
    Selection ();
    Sort population ();
    Elitism();
    Crossover {
    Create two populations:
    Population 1:  top 5-10% of the sorted population
    Population 2:  95-90% of the population, randomly picked
    Hooks-Jeeves Crossover() on Population 1
    Standard GA Crossover() on Population 2
    }
    Mutation ()
  }
}
```

3.1.2 Conjugate Direction GA Procedure

Similar to the Hooks-Jeeves hybrid GA procedure, an alternative crossover operator is developed based on Powell's method of conjugate direction to optimize along successive direction that are conjugate with respect to all previous directions [14]. Again, the new operator is applied to a subset of the population while the rest of the population undergoes the standard GA crossover operator. The main steps of the algorithm are described in the following pseudo code:

```
Conjugate Direction GA{
  while {termination criteria is not met
  …
    Crossover {
    Create two populations:
    Population 1:  top 5-10% of the sorted population.
    Population 2:  95-90% of the population, randomly picked
    Conjugate Direction Crossover() on Population 1
    Standard GA Crossover() on Population 2
    }
  ...
  }
}
```

3.2 Archived Noisy Genetic Algorithms

A GA-based noisy search method is developed to address real world problems, where uncertainty in the simulation model input parameters exists. In this approach, multiple realizations of the uncertain parameter are generated. Each generation of the algorithm uses a single realization, and the best solution for each generation is stored in a separate set called the "archive pool". The main steps of the algorithm are described below:

Step 1. Generate a number of realizations ($R_{1, 2, 3 \ldots rn}$), where *rn* is realization number, for the uncertain parameter.

Step 2. Initialize the GA by creating an initial population with population size (p_{gn}), where *gn* is generation number, for this step *gn = 1*.

Step 3. Run the simulation model to determine the calculated output. Only one realization is used per generation without replacement, i.e. a realization will be used at most one time then discarded, thus *total number of realization (TR) = total number of generation (TG)*.

Step 4. Evaluate the fitness of the calculated data from step 3 using the observed data. Several fitness error functions could be used, one of the common error function is root square error (RSE)

$$\text{RSE} = \sqrt{\sum_{i=1}^{n} (obs_i - cal_i)^2} \qquad (1)$$

Step 5. The most fit solution, i.e., the best individual *(BI)*, will be stored in the archive pool *(AP)*.

Step 6. The standard GA is applied to the "selection poll" to generate new p_{gn} ($gn > 1$) individuals. The selection pool (*SP*) contains $(p_{gn-1}+ar)$ individuals, where p_{gn-1} is the number of individuals in the previous population, and *ar* is the number of individuals in the archive pool.

Step 7. Repeat steps 3, 4, 5, and 6 until $gn = TG$. Note that the selection pool size increases as the generation increases. Thus, the selection population size is not a constant as in the standard GA.

The solution at the end of generation number *TG* will hopefully converge to an optimal solution for the problem. An alternative is to define a termination criterion for the procedure. This criterion could depend on number of realizations, sensitivity of the best solution or the improvement of the solution.

As explain above, the procedure utilized only one realization per generation. For more reliable and mature solution many

realizations could be used for a number of generations. Therefore some modifications are needed to enhance the procedure and make it more robust. For example, in ***Step3***, instead of one realization, a number of realizations *(NR)* could be used for a number of generations *(NG)*. For more conservative error function, Min Max error function could be used instead of RSE in ***Step 4***. As a result of these modifications, *BI* in ***Step 5*** should be stored in *AS* every *NG* generations.

4. SIMULATION MODEL EFFIECINCY IMPROVEMENT

Considering the computational burden of the S-O approach, two approaches are investigated to improve the execution time of the simulation model. The first approach is based on fine and coarse grained parallelism of the simulation model. The second is based on substituting the simulation model with fast surrogate models.

4.1 Simulation Model Parallelism

The simulation model used as an illustration in solving a groundwater inverse problem in this study is the Parallel Groundwater transport and REMediation codes (PGREM3D), a suite of massively parallel codes used for numerical simulation of three-dimensional groundwater transport and remediation problems [9]. These codes are based on the finite element methods (FEM). This transport module simulator is parallelized using a two-dimensional domain decomposition (in the x and y directions). Explicit message passing interface library (MPI) was utilized to exchange information between these domains. The codes are written in FORTRAN using double-precision arithmetic [9].

The simulation model is implemented to accommodate fine and coarse grained parallelism. Fine grained parallelism is applied by executing the simulation model on multiple computer processors via the MPI communication library [7]. The coarse grained parallelism is implemented using the MPI Group library, in which multiple instances of the simulation model are executed simultaneously in groups. In this technique, one processor P_0 acts as a model master while the remaining processors (*total number of available processors -1*) will be divided into a number of groups. Thus, each group consists of number of processors equal to *(total number of processors -1)/ (number of group)*. Within each group one processor acts as a group master to establish interprocessor communication with P_0 (Figure 1).

4.2 Surrogate Modeling

In addition to simulation model parallelism, surrogate modeling is being investigated to address the excessive computational intractability resulting from iterative evaluation of the simulation model. Improving simulation model efficiency could be addressed by constructing a surrogate model that is able to estimate quickly the simulation model output. A feed forward neural network-based procedure is used for constructing and training a surrogate model. As the level of complexity of a simulation model increases, the effort and resources needed to construct and train a surrogate model increase as well. This increase, however, occurs prior to the iterative search procedure, thus not contributing to the computational burden during the S-O procedure. In this study a preliminary investigation is done utilizing a two-dimensional numerical simulation groundwater transport and remediation code implemented in MATLAB to construct a surrogate model.

5. COMPUTATIONAL FRAMEWORK

5.1 Framework Architecture

This study is built upon the LAarge Scale Simulation-Optimization framework (LASSO) (Figure 2) [12]. LASSO consists of a centralized optimization application that utilizes a master-worker task distribution strategy. The optimization, master, and worker processes are executed on grid-based computational resources. The worker processes interface with instances of the forward model for distributed task execution. Results are returned to the master for processing by the centralized optimization application. In the following sections, key components of the application architecture are described in greater detail.

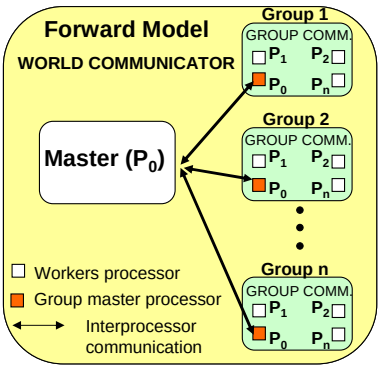

Figure 1. A Schematic of Simulation Model Parallelization

Several different search procedures have been implemented in Java, making up the centralized optimization application. Here, the optimization model representation of the inverse problem is solved using evolution strategies (ES) − a stochastic search heuristic conceptually similar to natural evolution [2]. An ES-based procedure encodes within an individual the decision variables that describe a potential solution to the inverse problem. The ES search starts with a collection of individuals, referred to as a population. The objective function (representing prediction error) of the optimization model is used to quantify a fitness value indicating how well an individual solves the inverse problem. Fitness values are calculated using the results of forward model evaluations. During the search process, the population is iteratively subjected to stochastic selection and mutation search operators. Each iteration of the algorithm constitutes a generation. This search process continues until a predefined convergence criterion is satisfied. The application of an ES-based search to inverse problems is advantageous because of their robustness and global search characteristics. Some drawbacks, however, include the computational intensity of a typical ES search and slow final convergence prior to termination.

5.2 Framework Parallelism

The optimization application is coupled with the forward model (PGREM3D) for fitness evaluation. Each forward model evaluation is handled by a number of processors (typically 1-8) called number of processors per group. The MPI is used to group

processors, associate processors to computational domains, and for fine grained message passing within each of these groups. The solution procedure adopted here involves three levels of parallelism: one level exhibited by the search procedure and the other two levels exhibited by the forward model. Each iteration of the search procedure exhibits a coarse grained parallel structure that requires an uncoupled forward model evaluation for each individual in the population. The optimization application acts as the master process in the master-worker task distribution strategy. The master, worker, and task pool used in the framework were designed and implemented as part of Vitri [3]. The master maintains a pool of remote tasks − a bundle of individuals requiring evaluation. Aggregating individuals in this manner reduces communications overhead. Worker processes running on distributed grid resources, having established a TCP-IP socket connection with the master, signal their readiness and draw tasks from the task pool [13]. The worker process transfers the remote task to the MPI zeroth processor. From this point forward, standard MPI group communications are utilized. The forward model manages multiple MPI groups, and each group evaluates an individual in the task bundle. The results of these simulations are then aggregated into a result bundle and returned to the optimization application for processing by the search algorithm. Finally, the next generation of the search is initiated.

The computational experiments presented here were performed on the NSF TeraGrid, NCSA. The TeraGrid is a heterogeneous agglomeration of computational resources distributed across the United States and connected through a specialized interconnection network designed for high-band width data transfer [6].

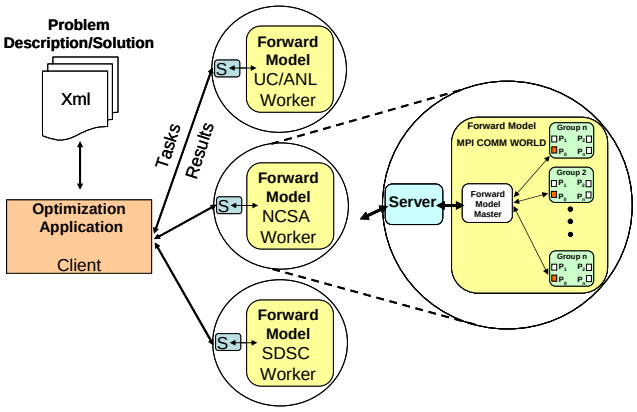

Figure 2. A Schematic of the Framework Architecture

6. APPLICATION DESCRIPTION AND EXPERIMENT SETTING

In this study a groundwater contaminant source characterization inverse problem is considered, where the forward model PGREM3D is employed for 3-dimensional problem while a 2-dimensional MATLAB version of the code is used for ongoing research investigation. The governing equations describing the groundwater transport are fully explained by [4] and [9].

6.1 Application Description

In the groundwater contaminant source identification (CSI) problem, an unknown contaminant release at a single source location is resolved from spatially and temporally distributed concentration observations collected at monitoring wells. The problem assumes that the contaminant source location and the contaminant release at the sources are unknown. Concentration observations at monitoring wells are collected to generate a concentration time-series at each monitoring location. The problem assumes that the source location and contamination release at the source are unknown. Furthermore, the signature of the source embedded in the monitoring data is a function of the source characteristics. For a three dimensional version of the problem, we attempt to model the observed concentration at the 18 monitoring wells (see the cross-sections of the domain in Figure 3) using PGREM3D, which describes the relation $\boldsymbol{Cm} = f(x_j, y_j, z_j, \boldsymbol{Cr})$, where x_j, y_j *and* z_j are the coordinates of the expected source location, $j = 1, 2$ denotes the vertices at opposite corners of the extent of the source, $\boldsymbol{Cr}$ is the source concentration, and $\boldsymbol{Cm}$ is the time series of modeled monitoring concentrations. For a two dimensional version of the problem, we model the observed concentration at the seven monitoring wells (Figure 3) using MATLAB version, which describes the relation $\boldsymbol{Cm} = f(x_j, y_j, \boldsymbol{Cr})$, where x_j, *and* y_j are the coordinates of the expected source location. The inverse problem is posed as an optimization model where the RSE between the observed and calculated concentrations (Equation 2) is minimized. The following constraints are included to enforce decision variable bounds and feasibility.

$$\min_{x_j, y_j, z_j, Cr} \sqrt{\sum_{i=1}^{n} (\mathbf{C}o_i - \mathbf{C}m_i)^2} \qquad (2)$$

Subjected to:

$$0 \le C_0 \le C_{max} \qquad (3)$$

$$0 \le x_j \le x_{max};\ 0 \le y_j \le y_{max};\ 0 \le z_j \le z_{max} \quad j=1,2 \qquad (4)$$

$$x_1 \le x_2;\ y_1 \le y_2;\ z_1 \le z_2 \qquad (5)$$

Depending on the number of contaminant sources, the number of decision variables is equal to the product of the number of contaminant sources times the number of unknowns used to describe each source. Thus, there are 7 and 5 unknown decisions variables and for 3D case and 2D case, respectively.

Table 1. Hypothetical Domain Parameters

Parameter	Values for 2-D problem	Values for 3-D problem
Problem size	51x31 grids	51x31x11 grids
Number of time steps	100	100
Time step size (dt)	2 day	0.15day
Grid spacing (dx, dy, dz)	2 m	2 m
Dispersion parameters αL, αT, Dm	1m, 1m, $0.01m^2/d$	0.1m,0.1m, $0.001m^2/d$
Velocity	1 m/day	1 m/day
True source location	x_1=8, y_1=30, x_2=12, y_2=34, C_0=70mg/L	x_1=2, y_1=15, z_1=6 x_2=5, y_2=17, z_2=8 8, C_0=70mg/L

6.2 Description of the Test Problem Domain

Hypothetical 2D and 3D fields were considered in this study. Detailed geometrical and hydraulic parameters are shown in Table 1, and in Figures 3 and 4.

6.3 Forward Model and Search Parameter Setting

For the 2D applications, the grid resolution for the simulation model resulted in 1,581 finite element nodes within the groundwater domain, and the simulation duration was 100 time steps, using 7 monitoring wells located close to the downstream end of the field (Figure 3). The population size for the MATLAB GA toolbox was 200 and the algorithm was executed for 100 generations. Other GA parameters and the boundary parameters used for this problem are shown in Tables 3 and 2, respectively.

Table 2. Allowable Range of Decision Variables Values for 2D and 3D Problems

Variables	Ranges
x axis	0-100
y axis	0-60
z axis	0-30
C_0	0-100

Table 3. GA Settings for the 2D Problem

Parameter	Setting for the MATLAB GA Toolbox
Population Size	200
Population Type	Double vector
Generation	100
Selection	Stochastic uniform
Crossover	Heuristic crossover, 0.8
Mutation	Gaussian mutation, 0.2

Table 4. ES Settings used for the 3D Problem

Parameter	Setting for the ES
Population Size	300
Population Type	Double
Generation	100
Sigma	3.0

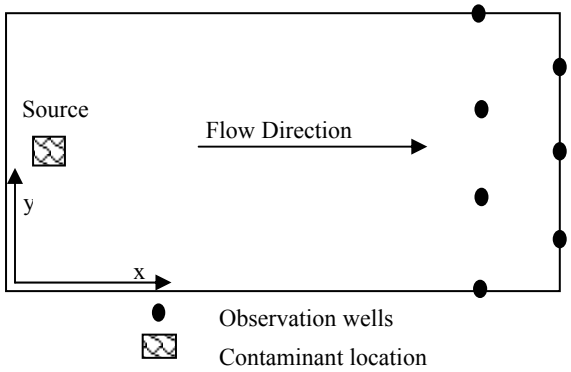

Figure 3. Hypothetical Two-Dimensional Domain

For the 3D applications, the grid resolution for the simulation model resulted in 17,391 finite element nodes within the groundwater domain, and the simulation duration was 100 time steps, using 18 monitoring wells located in the middle and farthest end of the field (Figure 4). The population size for the LASSO evolution strategies was set at 300 and the algorithm was executed for 100 generations. Other ES parameters and the boundary parameters used for this problem are shown in Tables 4 and 2, respectively. The best solution was found at a sigma setting of 3.0.

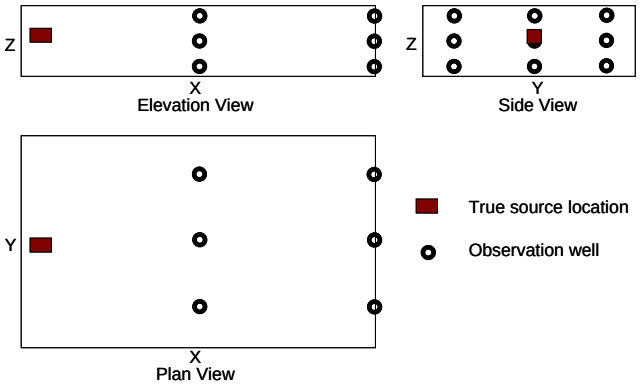

Figure 4. Hypothetical Three-Dimensional Domain

7. PRELIMINARY RESULTS

The results section consists of the application results and the framework performance results. In the application results, we report the computational framework results obtained for the 3D case. Also preliminary results of the Hooks-Jeeves hybrid GA procedure and surrogate model for the 2D case are reported.

7.1 Application Results

7.1.1 3D Problem Result using LASSO Framework

A set of runs were conducted on the TeraGrid NCSA site to address 3D inverse problems. Several trials were first conducted by tweaking the LASSO evolution strategies parameters, *population size, number of generations and sigma*. Figure 5 shows the true and the calculated values for the problem; noticeably the true and the estimated values are slightly different. The unknowns in the problem are the source location and concentrations, which make the problem more complex due to nonuniqueness. Based on deductions from the groundwater governing equations, the concentration has a linear trend while the source location has a nonlinear trend, which makes, in general, the estimation of source location more complex than concentration. While the RSE error is not fully minimized yet, improved sigma value could result in better solution performance since sigma is found to be more sensitive than the other parameters.

The number of forward model evaluations performed during a search is a function of population size times the number of generations. For each, we had a total of 30,000 evaluations distributed among 32 parallel forward simulation model groups with 2 processors per group (thus utilizing a total of 64 processors). On a normalized basis, an evaluation took around

0.0696 second, versus 1.6668 second on a single processor, i.e. solving the problem took around 35 minutes instead of 14 hours.

7.1.2 Hooks-Jeeves Hybrid GA Procedure Results

The new procedure was applied to the 2-dimensional source identification problem and the results are compared with those obtained via a standard GA. Table 5 shows the estimated parameters along with the objective function and solution errors for both GA and Hooks-Jeeves Hybrid GA. Using the same number of evaluations, the results illustrate that both algorithms are able to predict the source location and concentration with less than 5% error, and the Hybrid GA is able to find the solution with less than 1% error.

Table 5. Hooks-Jeeves Hybrid GA and Standard GA Results for the 2D problem

True Location	GA Toolbox	Hooks-Jeeves GA
x1 = 6	6.01	6
Y1 = 28	27.89	28
Y1 = 10	10.52	10
Y2 = 32	31.94	32
C_0 = 70	61.27	67.2
Objective Fun. Error	0.011	0.004
Solution Error	3.49 %	0.8 %

7.1.3 Neural Network-Based Surrogate Model Results

A surrogate model constructed and trained using neural network (NN) is compared with the 2D simulation model [8]. Both models were then used with a standard GA with same number of function evaluations to solve the 2D CSI problem. The GA parameters are shown in Tables 3. Table 6 shows that the overall quality of the solutions obtained with the two models are almost similar; however, the evolution time for the search method coupled with the simulation model is almost 2.5 longer than that required for the search coupled with the surrogate model. The result shows a significant success of neural network in predicting the simulation model accurately and relatively fast.

Table 6. Source Characterization Based on the Surrogate Model and Simulation Model for the 2D Problem

True Location	Simulation Model	Surrogate Model (NN)
x1 = 6	6.01	5.97
y1 = 28	27.89	27.79
y1 = 10	10.52	10.41
y2 = 32	31.94	32.22
C_0 = 70	61.27	60.73
Objective Function Error	0.011	0.023
Solution Error	3.49%	3.68%

7.2 Framework Performance Results

Additional sets of runs were conducted on the TeraGrid NCSA site to measure the performance of the current S-O framework in terms of fine and coarse grained parallelism. One set of runs was used to investigate fine grained parallelism within the FEM simulator. In this study, we determined wall time corresponding to increasing number of processors per group while number of groups and number of tasks remained constant. Figure 6 illustrates that evaluation time decreases until 4 processors per group then increases thereafter. That is because improvement in fine grained parallelism is associated with the problem size. As the problem size used is relatively small in this study, the improvement is limited.

The second set of runs was used to investigate coarse grained parallelism. Here we determine the evaluation time while we increased the number of groups. Other parameters such as number of evaluations, number of processors per group and number of tasks per group were kept unchanged. Theoretically, the application should scale linearly with the number of processors used. Scalability results shown in Figure 7 indicate that the framework scales almost identical to the theoretical scale until 32 groups, and then scales slight sub-linearly when the number of group exceeds 64. This could be due to the effect of communication that plays an important role in increasing the wall time when more than 32 groups are used.

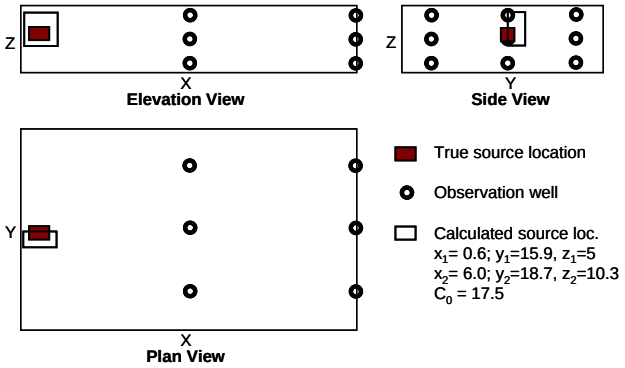

Figure 5. Estimated and True Source location for the Problem

8. FINAL REMARKS AND FUTURE WORK

This paper describes my dissertation research that is focused on addressing inverse problems using a S-O approach where the optimization is conducted using evolutionary algorithms. Methodological and computational investigations were performed to improve the time efficiency of the simulation component and the EA-based search procedures. The development of a distributed computational framework implemented on the TeraGrid site at NCSA is reported, where the solution procedure employs a distributed version of the EA-based search procedure. The preliminary results indicate that the effect of nonuniqueness in the solution for 3D contaminant source identification problem is significant. Grid-enabled parallelized framework was investigated in two ways: coarse grained within the population-based search algorithm; and fine and coarse grained within the simulation model. Overall, these implementations reduced the evaluation time drastically to minutes instead of hours, while maintaining the solution quality.

A set of preliminary results for the hybrid GA procedure and a neural network-based surrogate model was also presented. Preliminary results for the Hook-Jeeves hybrid GA method show that the new local-global search hybrid algorithm is able to achieve high-quality solutions for a 2D application problem. This investigation is being extended to the more complex 3D application problem. Results for solving an inverse problem,

which couples an evolution strategies-based search procedure with a 2D simulation model and a neural network surrogate model, shows identical solution quality, while the use of the surrogate model yielded significantly faster convergence.

Future work includes incorporating the embedded hybrid methods and the proposed procedure for a noisy GA into the parallel computational framework. These methods will be tested for a set of 2D and 3D groundwater contaminant source identification problems with varying degree of problem complexity. The surrogate modeling approach will be extended to 3D simulation cases, and then coupled with the search procedures to solve 3D inverse problems. Finally, this dissertation research plan intends to extend the framework to run in multiple TeraGrid sites to utilize both the fine and coarse grained parallelism to solve a highly complex groundwater source characterization problem with relatively more unknowns to resolve.

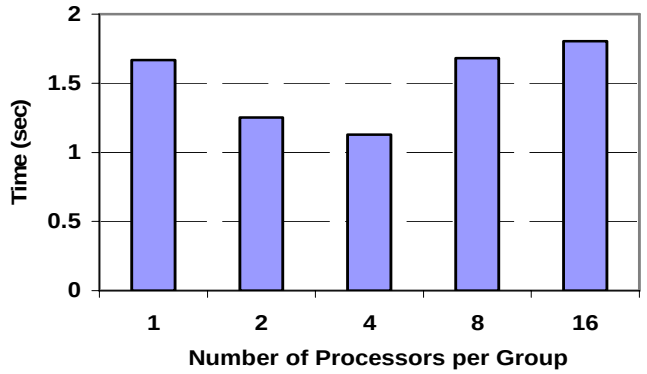

Figure 6. Fine Grained Parallelism, Number of Group =1, Task/Group=1:1

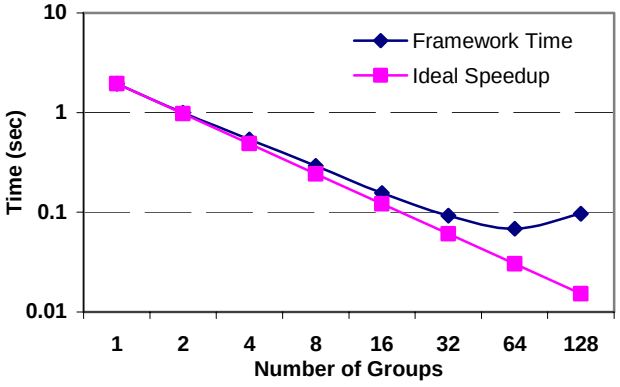

Figure 7. Coarse Grained Parallelism, Procs/Group =1:1, Tasks/Group= 1:1

9. ACKNOWLEDGEMENTS

This work was supported by National Science Foundation under Grant Numbers BES-0312841 and BES-0238623. The authors are grateful for the TeraGrid supercomputer resources provided by NCSA.

10. REFERENCES

[1] Atmadja, J. and Bagtzoglou, A.C. (2001), *"State of Art Report on Mathematical Methods for Groundwater Pollution source identification"*, Environmental Forensics, 2(3) 205-214, 2001b.

[2] Back, T. (1997), editor. *"Handbook of Evolutionary Computation"*, IOP Publishing Ltd. and Oxford University Press.

[3] Baugh, J.W. (2003), *"Vitri 2.0."*, Available: http://www4.ncsu.edu/ jwb/vitri/.

[4] Bear, J. (1979), *"Hydraulics of groundwater"*, New York: MCGraw-Hill.

[5] Belegundu, D.A. and Chandrupatla, R.T. (1999), *"Optimization concepts and applications in engineering"*, Prentice Hall Inc.

[6] Catlett, C. (2002), *"The TeraGrid: A primer"*, Available: http://www.teragrid.org/.

[7] Gropp, W., Lusk, W. and Skjellum, A. (1999), *"Using MPI: Portable Parallel Programming with the Message –Passing Interface"*, 2nd The MIT Press, Cambridge, MA.

[8] Johnson, V.M. and Rogers, L.L. (2000), *"Accuracy of neural network approximation in simulation-optimization"*, Journal of Water Resources planning and Management-ASCE, 126(2):48-56 Mar/Apr.

[9] Mahinthakumar, G. (1999), *"PGREM3D: Massively Parallel Codes for Groundwater Flow and Transport"*, Available: http://www4.ncsu.edu/~gmkumar/pgrem3d.pdf.

[10] Mahinthakumar, G. and Sayeed M. (2005), *"Hybrid genetic algorithm local search methods for solving groundwater source identification inverse problems"*, Water Resource. Plng. and Mgmt., vol. 131, no. 1, pp. 45–57, Jan/Feb.

[11] Miller, B.L. (1997), *"Noise, sampling and genetic algorithms"*, PhD Thesis, University of Illinois at Urbana-Champaign.

[12] Mirghani, B., Tryby, M., Ranjithan, R., Baessler, D., Nicholas, K. and Mahinthakumar, K. (2005), *"A Grid-enabled Simulation-Optimization Framework for Environmental Characterization Problems"*, Poster presentation at Super Computing Conf., Seattle WA.

[13] Mirghani, B., Tryby, M., Baessler, D., Karonis, N., Ranjithan, R. and Mahinthakumar K. (2005), *"Development and Performance Analysis of a Simulation-Optimization Framework on TeraGrid Linux Clusters"*, The 6th LCI International Conference, Chapel Hill, NC.

[14] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1996), *"Numerical recipes in Fortran"*, Second edition. Cambridge University Press, New York, NY.

[15] Sun, N.Z. (1994), *"Inverse problem in groundwater modeling, Theory and application of Transport in porous media"*, (Ed. Jacob Bear), Vol. 6, Kluwer Academic Publishers, 337p.

[16] Talbi, E.G. (2002), *"A taxonomy of hybrid metaheuristics"*, Journal of Heuristics Volume 8, P541-564.