Proceedings of the Annual International Conference on Soils, Sediments, Water and Energy

Volume 15 Article 11

June 2010

Using Genetic Algorithms on Groundwater Modeling Problems in a Consulting Setting

Karen M. Madsen
AECOM Technology, karen.madsen@groundwatergo.com

A. Elizabeth Perry
AECOM Technology, elizabeth.perry@aecom.com

Recommended Citation
Madsen, Karen M. and Perry, A. Elizabeth (2010) "Using Genetic Algorithms on Groundwater Modeling Problems in a Consulting Setting," *Proceedings of the Annual International Conference on Soils, Sediments, Water and Energy*: Vol. 15 , Article 11.
Available at: http://scholarworks.umass.edu/soilsproceedings/vol15/iss1/11

PART III: Modeling

Chapter 10

USING GENETIC ALGORITHMS ON GROUNDWATER MODELING PROBLEMS IN A CONSULTING SETTING

Karen M. Madsen[1§], A. Elizabeth Perry[2]
[1]*AECOM, 2 Technology Park Drive, Westford, MA 01886,* [2]*AECOM, 2 Technology Park Drive, Westford, MA 01886*

ABSTRACT

This paper presents a practical application for writing and applying simple genetic algorithms (GAs) for the common groundwater flow model, MODFLOW. The method employed by GAs is derived from the driving forces of evolution in the natural world. They employ functions that mimic natural evolutionary processes including selection, mutation, and genetic crossover. A GA solves mathematical problems where a desired outcome to the problem is defined (for example, calibration targets or remediation goals), but the inputs needed to arrive at this outcome are unknown. Our paper includes an introduction to genetic algorithms, the pseudocode of our genetic algorithm for MODFLOW, and the results of an experiential application. Due to the lack of commercially available GAs for MODFLOW, we coded a simple algorithm in Visual Basic Script and applied it to an example model. In the example model, the GA was used to conduct parameter estimation on a MODFLOW model of a river basin in New England that we had previously developed and calibrated in our practice. The calibration target used was net groundwater flow into the river. Four model input parameters were selected as chromosomes for the GA to act on: recharge, river conductance, and two general head boundaries. An initial population of 100 models was developed by varying the value of the gene parameters. The GA ran a MODFLOW simulation for each member of the population, extracted each output file, and established the error of each model from the calibration target. It then evolved the entire population of models towards the calibration target. The GA converged on a single set of input parameter that established best-fit values for all of the

[§] Corresponding Author: Karen Madsen, AECOM, 2 Technology Park Drive, Westford, USA, MA 01886, 617-780-9179, Email: karen.madsen@groundwatergo.com.

chromosome parameters. Genetic algorithms provide a practical alternative to trial-and-error and automated statistical calibration procedures, and can also be used for optimization.

Keywords: groundwater modeling, MODFLOW, parameter estimation, optimization, genetic algorithm.

1. INTRODUCTION

This paper presents a practical application for writing and applying simple genetic algorithms for MODFLOW. Such algorithms can be used for optimization and/or parameter estimation in groundwater modeling problems. The method presented is intended for environmental consultants who may tend away from using evolutionary algorithms in their practice due to the associated complexities and costs. Below we have presented a brief introduction to genetic algorithms, the pseudocode of our genetic algorithm for MODFLOW, and the results of an experiential application of this code to a parameter estimation problem.

1.1 Uses and Structure of Genetic Algorithms

The purpose of genetic algorithms (GAs) is to solve complex mathematical problems. GAs can be applied to almost any real world problem that can be structured numerically, from manufacturing supply-chain management to host/parasite relationships. Economic, legal, or political optimization can also be solved, as long as the problem can be constructed numerically (Mitchell and Taylor, 1999).

GAs are derived based on the driving forces of evolution in the natural world and include functions that mimic natural evolutionary processes including selection, mutation, and genetic crossover. A GA solves mathematical problems where a desired outcome to the problem is defined (for example, calibration targets or remediation goals), but the inputs needed to arrive at this outcome are unknown (input parameter values, numbers and locations of pumping wells, etc).

Genetic algorithms have many advantages over derivative-based optimization techniques, such as linear programming (Rizel and Eheart, 1994). They do not require continuity or convexity of the objective function (Espinoza et al., 2005). They are also free of the numerical difficulties that are frequently associated with derivative based optimizers (Wang and Zheng, 1997), and they can incorporate changing conditions and noisy observational data (Fogel, 2008).

In GA terminology, the term *member* refers to a set of trial input parameters. *Chromosomes* refer to each input parameter. When the chromosomes of a specific member are input into the problem and the problem is solved, the result will contain some degree of error when compared to the desired solution to the problem. Those members with less error are described as being *fitter* than those members with more error. *Population* refers to a collection a members. Populations are developed in a series of iterations, called *generations*. *Parent* population refers the ith iteration of populations and *child* population refers to the ith + 1 iteration (Mitchell and Taylor, 1999). Genetic algorithms allow the population to evolve over many generations until the population of resulting members converge on the desired solution. The evolutionary process includes these basic steps: 1.) an initial population is developed 2.) the fitter members of the population are selected as parents for the next generation 3.) mutation and genetic cross-over are conducted on the parent generation to create the child population, and 4.) steps 1-3 are repeated iteratively to move the entire population closer to the desired solution (Fogel, 2008).

An important step in any GA effort is the selection of GA operative parameters. These are differentiated from the chromosome parameters as follows: The chromosome parameters are inputs to the mathematical problem to be solved. The GA operative parameters are the parameters that control the GA mathematical algorithm. These include the population size, the number of generations, and the mechanism and frequency of cross-over, and mutation.

Since the 1970s researchers have been striving to identify a set of guiding principals that apply across all applications to establish equations for calculating operational parameters (Fogel, 2008). However, the work of D.H. Wolpert and W.B. Macready, *No Free Lunch Theorems for Optimization*, presented a proof showing that any possible guideline to setting operational parameter values is, by definition, problem-specific (Wolpert and Macready, 1997). That is, there are no universally applicable guidelines that can be used with all problems.

Although the No Free Lunch theorem indicates that absolute guidelines for setting parameter values cannot be established, it is certainly possible to set unsuccessful parameter values that will cause the GA to fail. For example, too large a mutation/crossover rate could cause the GA not to converge; too low a mutation/crossover rate could cause the GA to converge early; too small a population size could result in fewer solution sets being explored; etc. Another problematic phenomenon that has been identified is genetic drift, which occurs when mutation/crossover causes genes to fluctuate away from the best solution set and converge on non-optimal values. Reed et al. (2000) presented a method for setting certain GA operational parameters with the goal of avoiding genetic drift and poor solution set search for water resource problems. Such techniques may be helpful in establishing initial parameter values which can then be refined through trial and error.

Considering the still weak theoretic basis for GAs, perhaps the best method of determining that GA operative parameters have been set appropriately is that they result in convergence on a successful solution. As Back et. al. (1997) explained in their work *Evolutionary Computation: Comments on the History and Current State*, "We know that they work, but we don't know why." Thus the GA can be considered successful when it successfully locates input sets that satisfy the needs of the user.

In our application, trial and error were used to establish all GA parameters including the probability values for the rates of mutation and crossover. When a mutation occurred, its size was managed through a mutation scale factor, which represented the maximum percentage of the original value that could be added or subtracted. The coded equation for the mutated MODFLOW input parameter had the following structure:

$$P_n = P_o + P_o * R * F$$

P_n = new MODFLOW input parameter value
P_o = old MODFLOW input parameter value
R = random number between -1 and 1
F = the mutation scale factor (a constant)

Our work contains a noteworthy derivation from canonical GAs. We did not convert the gene parameters to binary strings. Historically, there has been a strong preference for mapping gene parameter values to binary strings prior to GA manipulation, the basis of which comes from GA schema theory. The schema theory, the roots of which go back to Holland, sought to characterize the underlying mathematical structure of GA search (Fogel, 2008). The theory stated that binary mapping would provide for more optimal sampling of the solution space (Back et al., 1997). More recent work has called schema theory into question, at least as it applies to many real world problems (Fogel, 2008). Binary coding may improve the performance of some models, but there are some known disadvantages to using binary strings. Primarily, that such coding may introduce additional multimodality, making the binary problem more complex than the original one (Back et al., 1997). For the purposes of this exercise, the disadvantage of mapping parameters to binary strings is more prosaic: it adds complexity to the writing, reviewing, and debugging of the code and, as our primary objectives are efficiency and cost effectiveness, this approach has not been employed. However, the method described in this paper could be easily modified to include mapping to binary strings.

Finally, an important step in GA development is placing appropriate constraints on the MODFLOW input parameters. In our application, when constraints were not controlled, the GA gravitated to unrealistic results, such as reversing the direction of flow in the river or converging on a water table 100 feet below sea level. In real world applications, parameters such as the elevation of the water table would probably be known within some narrow range. Placing tight constraints on MODFLOW input parameters, when appropriate, will limit the solution set such that only realistic values can be searched.

2. MATERIALS AND METHODS

Our impetus for this effort was the lack of commercially available GA codes for MODFLOW 2000. MODFLOW 2000 is one of the most popular groundwater modeling programs in existence, and thus is familiar to many consultants, their clients, and regulators (McDonald and Harbaugh, 1988; Winston, 1999). We chose to focus on MODFLOW 2000 because it is the program that we use most often in our own practice for large scale groundwater modeling problems.

We are not aware of any commercially available MODFLOW GUI software that includes a GA function. Because no commercial GA software is available, we also looked into open source GAs for MODFLOW. At the time of this writing, we are aware of one open source genetic algorithm for MODFLOW 1988, published by Chunmiao Zheng of the University of Alabama. This code is called MGO (Zheng and Wang, 2003). MODFLOW 1988 has been replaced in most consulting settings by MODFLOW 2000. A search was conducted for an open source GA for MODFLOW 2000, but a code was not found. Zheng and Patrick have written a genetic algorithm for MODFLOW 2000, called ModGA, which is owned by DuPont and is not commercially available (Zheng, 1997).

In the example presented in this paper, we developed a Visual Basic Script code to run a simple GA in conjunction with MODFLOW code to conduct parameter estimation. The presented code could be easily restructured for optimization problems. In order to apply the GA, the MODFLOW model must first be set up and roughly calibrated. Those input parameters designated as chromosomes must be identified and the range of reasonable values for these parameters must be established. A calibration target must be identified (for example, water table elevations could be used as the calibration target.)

The steps of our code are as follows:

1. The code is given the developed MODFLOW model and chromosome parameter values for the initial population.

2. The code reads the existing MODFLOW input files and duplicates those files for each population member while replacing the original chromosome parameters with unique chromosome parameter values for each member.

3. The code calls and runs MODFLOW for each population member.

4. The code reads the MODFLOW output files and calculates the error between each model's output and the user defined calibration target.

5. The code selects the fitter solutions using tournament selection; these fitter models become the parents of the next generation.

6. The code randomly conducts mutation and genetic crossover on some of parent models to create child models; other parent models advance into the next generation without mutation.

7. The entire process is repeated for the user-specified number of generations.

We used a simple naming convention for all MODFLOW files that included the generation number and the population member number in each name. This allowed for ease of coding, as the i and j integers from the do-loops were simply encoded in the file names. For ease of analyzing the results, each generation was saved in a unique folder.

The simple genetic algorithm for MODFLOW is presented in Figure 1 in Chapra and Canale's pseudocode (Chapra and Canale, 2002).

3. RESULTS AND DISCUSSION

3.1 GA Application

In our application, the GA was used to conduct parameter estimation on a MODFLOW model of a portion of a river basin in New England that had previously been developed in our practice and calibrated by trial-and-error methods. The model grid is shown in Figure 2. Four parameters were selected as the genes for the GA to act on: recharge, riverbed conductance, and two general head boundaries which define the upstream and downstream limits of the basin within the model. These were considered the parameters with the most uncertainty associated with them due to the inability to measure them in the field; other model parameters were kept constant. An initial population of 100 models was developed by varying the values of these chromosome parameters. The calibration target used in this application was net flow of groundwater into the river (calculated as the difference in streamflow between up and downstream USGS gauging stations). The GA code ran a MODFLOW simulation for each

```
DEFINE population size
DEFINE number of generation
DEFINE calibration target
DEFINE initial values of experimental parameters for first generation
DEFINE original MODFLOW model
DEFINE mutation and crossover factors
DOFOR i = 1 to number of generations
 DOFOR j = 1 to population size
'Create MODFLOW files for the population
  READ the original MODFLOW input files for non-chromosome parameters
  WRITE a copy of non-chromosome input files under the unique name of each
   member
  WRITE a copy of the MODFLOW NAM file for each member
  READ the original MODFLOW input files for chromosome parameters
  REPLACE the original chromosome parameter with the member specific
   chromosome value
  WRITE the member specific input files for chromosome parameters under the
   unique name of each member
'Run MODFLOW for each member of the population'
  CALL MODFLOW and run for each member
'Calculate each member's error'
  READ MODFLOW output file for each member
  IF member failed to converge THEN
   member error = non-convergence flag
  ELSEIF member converged
   member error = ABS(calibration target - member MODFLOW output)
  ENDIF
 ENDDO
'Conduct tournament selection to select fitter members'
 DOFOR k = 1 to half population size
  member a = RANDOM member of population
  member b = RANDOM member of population
  IF member a's error = non-convergence flag THEN
   IF member b's error = non-convergence flag THEN
    winner e = original model
   ELSE
    winner e = member b
  ELSE
   IF member b's error = non-convergence flag THEN
    winner e = member a
   ELSE
    IF member a's error <= member b's error THEN
     winner e = member a
    ELSE
     winner e = member b
    ENDIF
   ENDIF
  ENDIF
(Figure 1 continued on next page)
```

Figure 1. Pseudocode for the MODFLOW GA (continued on next page)

```
(Figure 1 continued from previous page)
(Repeat tournament selection for RANDOM members c and d and find winner f)
'Create child models from tournament winners through crossover and mutation.
'Conduct crossover on chromosome parameters when crossover factors apply.
'Crossover threshold and factor are set such that this is a rare occurrence.
    DOFOR m = 1 to number of chromosome parameters
      IF crossover factor * RANDOM < crossover threshold THEN
        experimental parameter m of child k = experimental parameter m of
          winner f
        experimental parameter m of child k + half population size =
          experimental parameter m of winner e
      ELSE
        experimental parameter m of new member k = experimental parameter m of
          winner e
        experimental parameter m of new member k + half population size =
          experimental parameter m of winner f
      ENDIF
    ENDDO
'Conduct mutation on chromosome parameters when mutation factors apply.
'Mutation factor 2 may be positive or negative.
    DOFOR n = 1 to number of experimental parameters
      IF mutation factor 1 * RANDOM < mutation threshold THEN
        experimental parameter n of new member k =
          experimental parameter n of new member k + RANDOM * mutation factor 2
      ENDIF
    ENDDO
(Repeat for mutation DO loop for new members k + half the population size)
  ENDDO
ENDDO
```

Figure 1. Pseudocode for the MODFLOW GA (continued from previous page)

member of the population, extracted each output file, calculated the groundwater discharge from each output file, and established the error of each model from the calibration target. Then, the GA selected the most fit solutions (closest to the calibration target) and developed the next generation. Repeating this process for several generations evolved the entire population towards the calibration target as described above until the all the members converged on a single set of input parameters.

Due to the simplicity of the experimental application described (i.e. only four genes were manipulated), the model rapidly converged until the error of the entire population was below an absolute percent error of 10%. After which, convergence continued more slowly. Figure 3 shows a graph of the absolute percent error versus the generation number. Each point on the graph is the error (different between model result and calibration target) for one member of the generation. Each generation includes 100 members. Thus at later generations, the error for all members of the population had converged on to a low level.

Running our GA application for a population size of 100 members for 50 generations took approximately 11 hours of computation time on a desktop PC. For the purposes of observing the evolutionary progress, all input and output files were saved. These files represented a memory burden of 260 MB. The GA could be programmed to delete files after use as needed based on memory constraints.

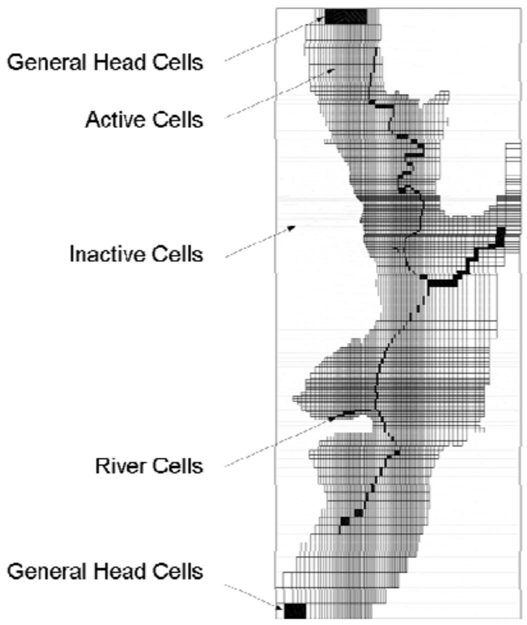

Figure 2. Model Grid

3.2 History and Present State of Genetic Algorithms in Water Resource Problems

This section briefly describes the history and current state of genetic algorithms as the relate to water resource problems. Genetic algorithms were first conceptualized in 1962 by J. H. Holland in his work *Outline for a Logical Theory of Adaptive Systems* (Holland, 1962). However, due to the expense and limited capacity of the computers at the time, GAs were not widely used until the 1970s. In the 1980s, improvements in computer technology allowed practitioners to apply GAs and other evolutionary algorithms to real world optimization problems. Their application became so common in the 1980s that by 1985, international conferences began being held on the subject (Back et. al., 1997).

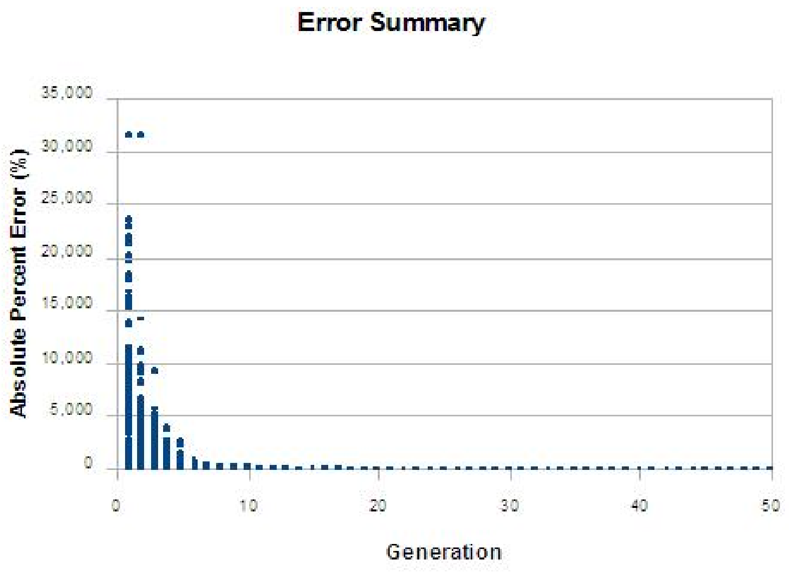

Figure 3. Error Summary – Absolute Percent Error versus Generation Number

For more than a decade, evolutionary algorithms have been applied to a wide variety of groundwater modeling problems. They are especially useful for the placement of wells in pump-and-treat well-field remediation systems, due to large possible solution set for these problems and the difficulty of testing all possible variations by hand (Chang and Hsiao, 2002). They have also been applied in parameter estimation in water resource problems (Kalwij and Peralta, 2006).

While research continues to advance on this subject, the practical application of these techniques is still hampered by resource limitations (Johnson and Rogers, 1995). These practical constraints are mainly associated with computational limitations, (Wang and Zheng, 1997) but they also include the human effort involved in properly setting up and coding the problem.

The uses of GAs are very broad in groundwater modeling applications, but they can generally be categorized as either parameter estimation or optimization problems (Babbar and Minsker, 2006; Tsai et. al., 2003).

A simple GA is best suited to finding a single set of input parameters that result in the desired solution to a problem, as the interaction between population members causes the chromosomes of the all members in a population to cluster around a single value. However, complex GA protocols have been developed which find multiple chromosome sets. Research continues to be conducted on so-called niche GAs. The theory of niche GAs is metaphorically parallel to evolutionary niches that occur in nature, where populations are isolated from each other and evolve different mechanisms for solving the same problems. Some examples of work that has been conducted in this area include Vector Evaluated GAs (Schaffer, 1985) and Niched Pareto GAs (Horn et al., 1994).

In their work *Massive Multimodality, Deception, and Genetic Algorithms*, Goldberg et al. (1992) set up an experimental function with over five million local optima and 32 global optima. They showed that a simple GA could find one of the global optima if the correct GA operational parameters were used. They also developed a niched GA that successfully located all 32 global.

All of these trends indicate the advances and likely long-term utility of GAs in water resources applications in the future.

4. CONCLUSION

In conclusion, genetic algorithms provide a practical alternative to trial-and-error and automated statistical calibration procedures, and can also be used for optimization. Here we have presented a practical method for writing and applying simple genetic algorithms for MODFLOW to be used in optimization and/or parameter estimation in groundwater modeling problems. Genetic algorithms have advantages over other commonly-used parameter estimation and optimization methods. It is hoped that this example increases awareness of the availability of these methods, demonstrates the use of GAs in a non-academic setting, and encourages further such applications in the future.

6. REFERENCES

Babbar, M. and Minsker B.S. 2006. Groundwater Remediation Design Using Multiscale Genetic Algorithms. Journal of Water Resources Planning and Management. September/October, 341-350.

Back, T., Hammel U., and Schwefel H. P. 1997. Evolutionary Computation: Comments on the History and Current State. Transactions on Evolutionary Computation. 1, 1, 3-17.

Chang, L-C. and Hsiao C-T. 2002. Dynamic Optimal Ground Water Remediation Including Fixed and Operation Costs. Ground Water. 40, 5, 481-490.

Chapra, S. C. and Canale, R. P. 2002. Numerical Methods for Engineers. McGraw Hill. Forth Ed, 28-39.

Espinoza, F. P., Minsker, B.S., and Goldberg, D. E. 2005. Adaptive Hybrid Genetic Algorithm for Groundwater Remediation Design. Journal of Water Resources Planning and Management. January/February, 14-24.

Fogel, D. B. 2008. Introduction to Evolutionary Computation. In: Modern Heuristic Optimization Techniques. pp. 3-23. (K. Y. Lee and M. A. El-Sharkawi) Institute of Electrical and Electronics Engineers.

Goldberg, D. E., Kalyanmoy, D., and Horn, J. 1992. Massive Multimodality, Deception and Genetic Algorithms. In: Parallel Problem Solving from Nature. pp 37-46. (R Manner and B Manderick) Amsterdam: North-Holland.

Holland, J. H. 1962. Outline for a Logical Theory of Adaptive Systems. Journal of the Association for Computing Machinery. 9, 3, 297-314.

Horn, J., Nafpliotis, N., and Goldberg, D. E. 1994. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. World Congress on Computational Intelligence. 1, 82-87.

Johnson, V. M. and Rogers, L. L. 1995. Location Analysis in Ground-Water Remediation Using Neural Networks. Ground Water. 33, 5, 749-758.

Kalwij, I. M. and Peralta, R. C. 2006. Simulating/Optimization Modeling for Robust Pumping Strategy Design. Ground Water. 44, 4, 574-582.

McDonald, M. G. and Harbaugh, A. W. 1988. A Modular Three-Dimensional Finite-Difference Ground-Water Flow Model. In: Techniques of Water-Resources Investigations of the United States Geological Survey. Chapter A1. Department of the Interior. Denver, CO. USGS Open File Report 83-875.

Mitchell, M. and Taylor, C. E. 1999. Evolutionary Computation: An Overview. Annual Review of Ecology and Systematics. 20, 593-616.

Reed, P., Minsker, B., and Goldberg, D. E. 2000. Designing a Competent Simple Genetic Algorithm for Search and Optimization. Water Resources Research, 36, 12. 3757-3761.

Rizel, B. J. and Eheart, W. J. 1994. Using Genetic Algorithms to Solve a Multiple Objective Groundwater Pollution Containment Problem. Water Resources Research. 30, 5, 1589-1603.

Schaffer, J. D. 1985. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Proceedings of the First International Conference on Genetic Algorithms and their Applications, 93-100.

Tsai, F. T-C., Sun, N-Z, and Yeh, W. W-G. 2003. A Combinatorial Optimization Scheme for Parameter Structure Identification in Groundwater. Ground Water. 41, 2, 156-169.

Wang, M., and Zheng, C. 1997. Optimal Remediation Policy Selection under General Conditions. Ground Water. 35, 5, 757-764.

Winston, R. B. 1999. MODFLOW-Related Freeware and Shareware Resources on the Internet. Computers and Geosciences. 25, 377-382.

Wolpert, D.H. and Macready, W.B. 1997. No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation. 1, 67-82.

Zheng, C. 1997. ModGA: A Genetic Algorithm Based Groundwater Flow and Transport Optimization Model MODFLOW and MT3D, Report to DuPont Company, Hydrogeology Program, University of Alabama.

Zheng, C., and P.P. Wang. 2003. MGO: A Modular Groundwater Optimizer incorporating MODFLOW and MT3DMS; Documentation and User's Guide, The University of Alabama and Groundwater Systems Research Ltd.