WATER RESOURCES RESEARCH, VOL. 36, NO. 12, PAGES 3757–3761, DECEMBER 2000

Designing a competent simple genetic algorithm for search and optimization

Patrick Reed and Barbara Minsker
Department of Civil and Environmental Engineering, University of Illinois, Urbana

David E. Goldberg
Department of General Engineering, University of Illinois, Urbana

Abstract. Simple genetic algorithms have been used to solve many water resources problems, but specifying the parameters that control how adaptive search is performed can be a difficult and time-consuming trial-and-error process. However, theoretical relationships for population sizing and timescale analysis have been developed that can provide pragmatic tools for vastly limiting the number of parameter combinations that must be considered. The purpose of this technical note is to summarize these relationships for the water resources community and to illustrate their practical utility in a long-term groundwater monitoring design application. These relationships, which model the effects of the primary operators of a simple genetic algorithm (selection, recombination, and mutation), provide a highly efficient method for ensuring convergence to near-optimal or optimal solutions. Application of the method to a monitoring design test case identified robust parameter values using only three trial runs.

1. Introduction

Simple genetic algorithms (GAs) have been used successfully for a number of engineering design applications within water resources [see *Wang*, 1991; *Ritzel et al.*, 1994; *McKinney and Lin*, 1994; *Cieniawski et al.*, 1995; *Aly and Peralta*, 1999]. A principal difference between optimization using a GA versus more traditional methods is that the decision space is searched from an entire population of potential designs. This difference enables GAs to solve discrete, nonconvex, discontinuous problems without differentiation [*Goldberg*, 1989]. A major difficulty when using a GA for optimization lies in the large number of parameters that must be specified to control how the decision space is searched. The practitioner must adequately size the population of potential designs and set the selection pressure, the probabilities of recombination (mating) and mutation, and the run length (or number of generations) such that the GA effectively searches the decision space for a given problem. Setting these parameters in water resources applications to date has been a time-consuming trial-and-error process. *Aly and Peralta* [1999], for example, used 60 runs to identify appropriate parameter values.

The purpose of this technical note is to present existing tools from the genetic algorithm literature that can vastly limit the number of parameter combinations that must be considered to ensure that a GA converges to optimal or near-optimal solutions. Relationships for population sizing, convergence rates, and genetic drift are presented and demonstrated within the context of a long-term groundwater monitoring application. The goal of the application is to minimize long-term monitoring costs using the methodology developed by *Reed et al.* [this issue].

Paper number 2000WR900231.
0043-1397/00/2000WR900231$09.00

2. Basics of a Simple Genetic Algorithm

A GA searches a decision space using a process that is analogous to Darwin's "natural selection." The decision variables associated with the optimization model to be solved are encoded as 0-1 binary strings or chromosomes. The fitness of each member of a randomly generated initial population of these strings is determined by a penalty-based objective function developed for the optimization model of interest. After each individual is assigned a fitness, the GA finds optimal solutions using three basic operators: (1) selection, (2) crossover (mating), and (3) mutation. Initially, strings are selected at random, and those who possess a higher relative fitness are then allowed to enter the mating population.

A number of selection methods exist, but this paper assumes that tournament selection will be used. *Goldberg and Deb* [1991] showed that tournament selection is more efficient and less prone to premature convergence relative to other selection methods. Only the convergence rate relationships discussed in this paper are dependent on the selection scheme used. Practitioners interested in using selections schemes other than tournament selection should reference *Thierens and Goldberg* [1994] and *Thierens et al.* [1998] to attain the appropriate convergence rate relationships.

Tournament selection allows only the fittest individual from a group of strings randomly drawn from the current population to be placed in the mating population. Next, the crossover operator couples members of the mating population to mate with a specified crossover probability (P_c). Mating consists of randomly selecting one or more crossover points at which the strings exchange bit values (genes) with each other. If mating does not take place, the parents are replaced in the population. Selection and crossover are repeated until a specified population size is reached. Last, mutation occurs when random genes within the new population are changed with a given probability of mutation (P_m). These three operators act to create a new population (or generation) of individual sampling plans with

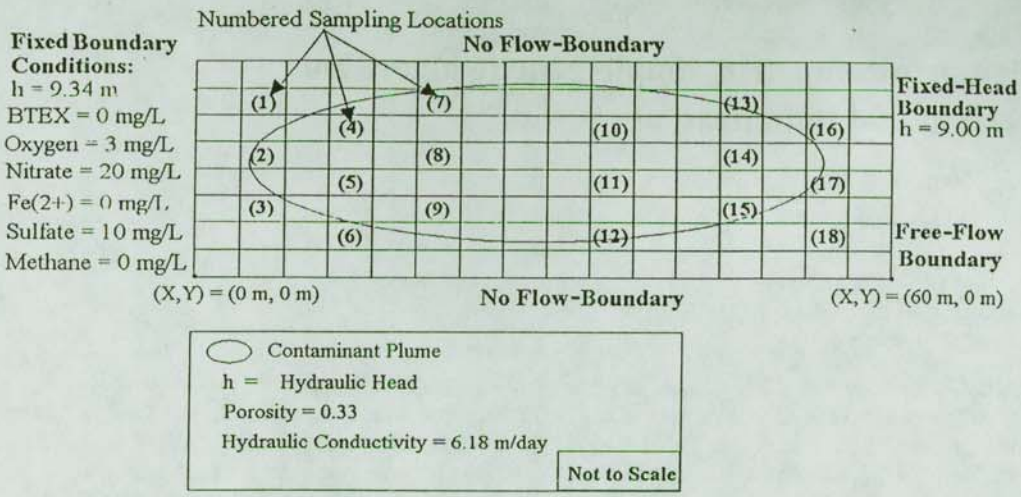

Figure 1. Plan view schematic of the modeled domain.

improved average fitness. Convergence to optimal or near-optimal solutions occurs when a single solution dominates the new population. This condition occurs when a user-specified percentage (e.g., 90%) of the strings composing the new population have bit values equal to those of the dominant solution. The schema theorem is the general theory describing how these three operators combine to evolve high-quality near-optimal solutions (see *Goldberg* [1989] or *Holland* [1975] for more information). It states that highly fit strings are composed of small chunks of information (or building blocks) that are relevant to the solution of the problem. The GA exerts a selection pressure where only highly fit members are allowed to pass their traits or building blocks to the next generation. Highly fit parent strings are allowed to mate, yielding offspring that inherit building blocks from both parents. It is in this manner that the GA assembles optimal or near-optimal solutions to a problem.

3. Monitoring Application

The application used to demonstrate GA performance in this paper minimizes long-term monitoring costs while accurately quantifying the mass of dissolved xylene (BTEX) predicted to be present at the site by reactive transport in three dimensions (RT3D). The test case used in this study consists of a hypothetical site with a 60-m by 20-m homogeneous and isotropic sandy aquifer contaminated with benzene, tolulene, ethyl benzene, and xylene. An existing reaction package within RT3D was used to model the multispecies transport and biodegradation of BTEX for multiple electron acceptors (for details see *Clement et al.* [1998]). Figure 1 is a plan view schematic of the modeled domain. Figure 1 shows the hydraulic flow and contaminant transport boundary conditions used in this study. RT3D's role in this application is to predict contaminant concentrations in a future sampling period at all existing monitoring well locations.

The goal of the optimization is to identify subsets of these well locations that accurately describe the global contaminant mass in the plume at minimal costs. Each sampling plan (or population member) is represented by a binary string (or chromosome). Within each of these chromosomes a value of 1 in the ith digit represents sampling from the ith well at the site. The fitness of each member of the population is determined by a penalty-based objective function that includes cost and accuracy of the global mass estimate. Further details on the methodology are given by *Reed et al.* [this issue].

4. Adaptive Search: Steps Ensuring High-Quality Solutions

Three steps are outlined below for identifying appropriate parameter values for the GA; the steps are tested using the monitoring application described in section 3.

4.1. Step 1: Preliminary Problem Analysis

The first step in designing a competent GA is to perform some basic calculations to identify a potential range of population sizes and examine the feasibility of solving the problem as it is currently formulated using a GA. The population size N can be estimated using (1), a population sizing equation for GAs developed by *Harik et al.* [1997], which considers "... two factors that influence convergence quality: the initial supply of building blocks (BBs) and the selection of the best BBs over their competitors." Equation (1) is based on a random walk model that represents the supply and interaction of competing BBs as a one-dimensional gambler's ruin problem:

$$N \geq -2^{K-1} \ln(\alpha)(\sigma_{bb}\sqrt{\pi(m-1)}/d). \quad (1)$$

The BB order K represents the minimum number of binary digits that have physical significance to solution of the problem. To insure that the solution converges to a near-optimal solution with a certainty greater than 95%, α (the probability of failure) should be set to less than 5%. The variable σ_{bb} represents the standard deviation of BB fitness, and d is the signal difference between the best and second-best building blocks. The final term, m, represents the maximum number of BBs within a single string. The term $\sigma_{bb}\sqrt{\pi(m-1)}$ represents the noise interference between competing BBs.

Equation (1) is a useful analytic tool for describing population size as a function of BB abundance and interaction, but often the practitioner in engineering applications cannot directly identify BBs and hence cannot directly estimate the parameters in (1). The order, standard deviation, and number of building blocks must then be estimated based on conservative assumptions and available data. The initial parameter that

must be estimated is the BB order K, which represents the smallest number of fixed positions within a binary string that have relevance to the problem being solved. The BB order can be conservatively estimated to range from 1 to 5 for most applications. *Goldberg* [1989] shows that higher-order BBs have a low probability of surviving selection, crossover, and mutation.

The noise interference between competing BBs, $\sigma_{bb}\sqrt{\pi(m-1)}$, can be approximated using the fitness standard deviation as shown in (2) [*Harik et al.*, 1997]:

$$\sigma_{bb}\sqrt{\pi(m-1)} \approx \sigma_{\text{fitness}}. \tag{2}$$

Here σ_{fitness} can be estimated from the standard deviation of the fitnesses attained from a randomly generated population of initial designs. For the monitoring application presented in this study, σ_{fitness} was found to equal 3090 by taking the standard deviation of the fitness values for 1000 individuals within a randomly generated trial population. This is a conservative estimate of the noise between competing BBs because for each succeeding generation, the GA is acting to converge the population toward a single solution, thereby reducing the standard deviation of the fitness values (i.e., reducing population size estimates).

The parameter d (the signal difference) should be assigned the value of the smallest fitness difference between competing individuals that the GA should be able to resolve. For the monitoring application several plans exist that have the same cost and must be differentiated by the accuracy of their mass estimates. For example, there are 18,564 sampling designs that sample 12 of the 18 wells. Each of these designs has the same cost of $4200. The GA differentiates between these plans using their respective penalty function values, which are a function of the accuracy of each plan's mass estimate. For this study, d was set equal to 200, which is the amount the penalty function changes for a 1% change in the accuracy of global mass estimates for BTEX. Any change below this value was assumed to be insignificant for discriminating between different fitness values.

Given these estimates for each of the parameters of (1), simple hand calculations can provide the user with a range of potential population sizes varying as a function of BB order K. For this study, population size estimates ranged from 46 to a maximum of 745 for the range of BB orders between 1 and 5. If population size estimates are infeasibly large, then the user should assess whether constraint violations are being excessively penalized. Excessive penalties for constraint violations can greatly influence the magnitude of the fitness standard deviation used to estimate the population sizes. This circumstance should be avoided because the GA's performance can be adversely affected. Excessive penalties greatly narrow the region of the decision space searched by a GA and can lead to suboptimal solutions. Solutions that are slightly infeasible often contain BBs that are important in attaining an optimal solution. Prohibitively penalizing infeasibilities quickly removes these solutions from successive generations, causing the loss of potentially useful traits (or BBs).

Once a range of population sizes is estimated as described above, the computational complexity of the GA can be analyzed. The computational complexity of a GA is measured as the number of function evaluations that are required to attain an optimal solution. The number of function evaluations can be calculated by multiplying the GA's population size (N) by the number of generations required for convergence (t).

The number of generations required for convergence is strongly affected by the relative rates that the genes (the binary digits that compose a single string) within the population converge. Two extreme cases have been modeled: (1) where all of the genes are equally important to the final solution, resulting in uniform convergence rates for the genes within the population, and (2) where the importance of each gene varies as an exponential function of that gene's location within a string, causing highly disparate convergence rates for each of the gene locations. These two cases represent the bounds for the convergence rates that will be encountered for GA applications, which have been found to be functions of $O(\sqrt{l})$ and $O(l)$ for cases 1 and 2, respectively (assuming tournament selection) [*Thierens and Goldberg*, 1994; *Thierens et al.*, 1998], where l is the string length. Most engineering applications have BBs that converge at different rates, a condition represented by specific genes converging at variable rates within the population. This phenomenon has been termed "domino convergence" [*Thierens et al.*, 1998] because BBs blocks converge sequentially, which is analogous to a series of dominos falling into place. Equation (3) estimates the expected number of generations (t) required under domino convergence for all string locations to be converged, assuming binary tournament selection [*Thierens and Goldberg*, 1994; *Thierens et al.*, 1998],

$$t \approx 2l. \tag{3}$$

Equations (1) and (3) provide a simple means to attain a preliminary assessment of the number of function evaluations that will be required to solve a problem. For this study, the string length (l) equals 18, and the resulting number of function evaluations needed to attain an optimal solution ranges between 1700 and 27,000 depending on the BB order used in (1). Each function evaluation requires approximately 0.12 s, so the problem should be solved within 3.4 to 54 min, a very feasible range.

The computational complexity of the GA can also be assessed in more general terms. Note from (1) that population size is directly proportional to m (or the number of BBs within a string) and that the number of BBs within a single string is proportional to that string's length (l). The required population size to attain an optimal solution grows as a function of $O(l)$, and recall that the convergence times range between functions of $O(\sqrt{l})$ and $O(l)$ (assuming tournament selection). Using these relationships, the computational complexity of a binary coded GA falls within a range of $O(l^{1.5})$ to $O(l^2)$, a relatively low value given that the decision space grows as the function 2^l. If these initial analyses indicate that the problem cannot be solved within a reasonable period of time, then the user should explore alternative solution methods or simplify the problem until it can be solved.

4.2. Step 2: Trial Runs and Parameter Settings

The tasks that remain after the preliminary problem analysis are specifying the appropriate population size and the probabilities of crossover and mutation. Recommendations for setting P_c and P_m result from a considerable body of prior research on GA performance as a function of these parameters. *Thierens* [1995] investigated the effects that selection and crossover have on solution quality and defined an upper bound on P_c for tournament selection shown in (4):

$$P_c \leq (s-1)/s. \tag{4}$$

The parameter s represents the total number of individuals that compete in tournament selection. Equation (4) is intended

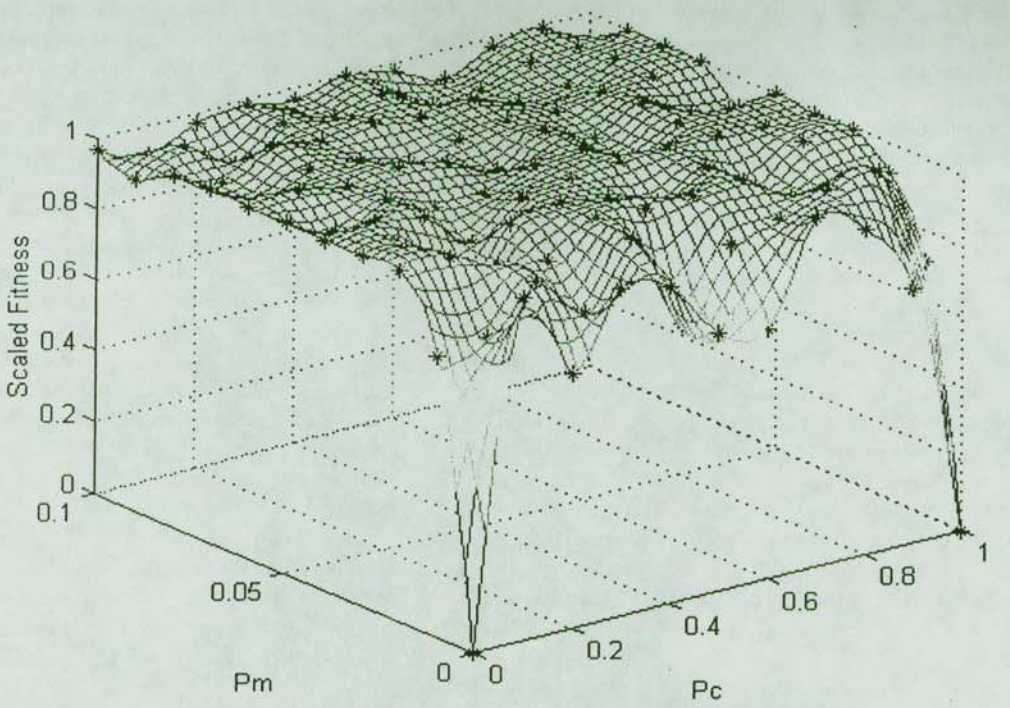

Figure 2. Trial run results for varying the probabilities of crossover and mutation.

to protect pertinent building blocks from being destroyed because of excessive mating. The long-term monitoring application presented in this paper used binary tournament selection with s equal to 2. The resulting upper bound for the probability of crossover is equal to 0.5. *Thierens and Goldberg* [1993] show that crossover exerts an innovative force on a system by combining the decision variables of only the fittest designs. This force of innovation should be maximized by using the largest P_c possible below the upper bound defined by (4), which predicts solution instability for P_c near and above 50%.

Equation (4) does not account for the effects of mutation, which results in relatively local refinements of solutions. *De Jong* [1975] showed that GA performance is generally acceptable when P_c is high and P_m is set as the inverse of the population size, as shown in (5):

$$P_m \approx 1/N. \tag{5}$$

The question that remains to be answered is which of the potential population sizes calculated in step 1 should be used. This question can be answered by examining the robustness of solutions attained for increasing values of the BB order (K). A good rule of thumb is to set the probabilities of crossover and mutation using relationships (4) and (5), assume a low value for K, perform a trial run, then increase K and perform a second trial run. If the results are similar, then the smallest population size should be implemented to minimize the number of function evaluations required to arrive at an optimal solution. If the results vary significantly, then continue to increase K until the solutions are similar. Given that K usually varies only from 1 to 5 and is an integer value, this approach will require relatively few runs.

For the monitoring application, if K equals 1 or 2, then population sizes of 50 or 100, respectively, are recommended. Setting P_c equal to 0.4 and P_m equal to 0.01, the trial run results yielded scaled fitness values equal to 0.65 and 0.99 for the population sizes 50 and 100, respectively. The fitness values are scaled such that a value equal to 1 represents the best solution attained for all trial runs performed in this study. All solutions with a scaled fitness greater than 0.9 minimize costs and have mass estimation errors that range up to a maximum of 1%. This result shows that a population size of 50 is too small and at least one additional trial run is necessary.

A population size of 200 was used for the additional trial run, which is slightly greater than the recommended value when K equals 3. The trial run yielded a scaled fitness value of 0.95 for this population size using the same probabilities of crossover and mutation as before. Hence a population size of 100 was judged to be most appropriate of the initial population estimates calculated in step 1. In this study, population sizes above 200 were not considered because each generation would require more function evaluations. Moreover, the total number of generations required before the GA converges would also increase significantly at larger population sizes. Although larger population sizes can often find optimal or near-optimal solutions in earlier generations because of the increased number of individuals that are considered per generation, these solutions cannot be identified as optimal solutions until they dominate the population, which takes longer with a larger population.

Figure 2 maps the GA's performance as a function of P_c and P_m when the population size is set equal to 100. The upper bound for P_c presented in (4) accurately predicts the instabilities in the GA's performance when P_m is very near or equal to zero. Interestingly, Figure 2 shows that the GA's performance is stable for most P_c values when P_m is greater than or equal to 0.02 (or approximately twice the value recommended by *De Jong* [1975]). Generally, the trial runs for this study show that increasing P_m by a factor of 2 or 3 above the value attained from (5) can avoid performance instability for all ranges of P_c when the population size is adequate. Figure 2 shows that the theoretical relationships presented in steps 1 and 2 can be used to attain control parameter values that yield robust results.

4.3. Step 3: Analysis for Drift Stall

A final phenomenon that must be considered in conjunction with domino convergence is "genetic drift" [*Thierens et al.*, 1998]. Genetic drift occurs in a population when crossover and mutation cause genes to fluctuate and converge to nonoptimal values in the absence of selection pressure [*Thierens et al.*, 1998]. Domino convergence causes genes within individual strings to experience varying selection pressures based on their relevance to the final solution. Genes with reduced relevance to the solution experience reduced selection pressures (or convergence rates). Although these genes are not converging because of selection, they may converge to nonoptimal values under the operations of crossover and mutation. Equation (6) gives the expected number of generations for genes to converge in the absence of selection within a randomly generated initial population consisting of binary strings with equal proportions of ones and zeros [*Thierens et al.*, 1998]:

$$t_{\text{drift}} \approx 1.4N. \tag{6}$$

Equation (6) shows that the expected time for convergence due to genetic drift is a linear function of the population size (N). Equations (3) and (6) can be combined to ensure that domino convergence to optimal solutions occurs before genetic drift can occur, as shown in (7):

$$t < t_{\text{drift}}. \tag{7}$$

If (7) is not satisfied, then genetic drift will cause the GA to converge to a nonoptimal solution, a condition that is termed drift stall. The population size should be increased until (7) is satisfied. The monitoring application presented in this study has a timescale for domino convergence equal to 36 generations, which is less than the 140 generations representing the timescale for genetic drift when $N = 100$. Equation (7) is particularly important for applications with large string lengths. As the number of decision variables increases, the timescale for domino convergence grows, a condition that promotes the likelihood that genetic drift will take place. Equation (7) can be used to refine the population size estimates resulting from steps 1 and 2 to insure that drift stall does not occur.

5. Conclusion

This paper presents a simple three-step method that the practitioner can use to limit the number of control parameter combinations that must be tested to ensure that a GA is converging to optimal or near-optimal solutions. The three steps present basic theoretical relationships describing the performance of a GA as a function of its control parameters. Step 1 of the methodology provides valuable information on the potential computational effort required to solve a given application, as well as allowing the practitioner to assess the effects that penalty functions have on the GA's performance. Step 2 allows the user to identify parameter settings for the population size and the probabilities of crossover and mutation using theoretical relationships available in the genetic and evolutionary computation literature and a limited number of trial runs. Step 3 allows the practitioner to refine population estimates attained from the previous steps to ensure that problems with large numbers of decision variables do not encounter drift stall. The three-step procedure is demonstrated using a monitoring design test case. Effective parameter values were identified with only three trial runs, which is far less than the number of runs required to identify parameter values using the trial-and-error methods.

Acknowledgments. This work was supported by the United States Geological Survey, the Illinois Waters Resources Center, and the United States Environmental Protection Agency's Science To Achieve Results (STAR) graduate fellowship program. Pam Yuen is gratefully acknowledged for performing the test runs discussed in this technical note. The comments of two anonymous reviewers, which greatly improved the quality of this paper, are also acknowledged.

References

Aly, A. H., and R. C. Peralta, Comparison of a genetic algorithm and mathematical programming to the design of groundwater cleanup systems, *Water Resour. Res.*, *35*(8), 2415–2425, 1999.

Cieniawski, S. E., J. W. Eheart, and S. Ranjithan, Using genetic algorithms to solve a multiobjective groundwater monitoring problem, *Water Resour. Res.*, *31*(2), 399–409, 1995.

Clement, T. P., Y. Sun, B. S. Hooker, and J. N. Petersen, Modeling multispecies reactive transport in ground water, *Ground Water Monit. Rem.*, *18*(2), 79–92, 1998.

De Jong, K. A., An analysis of the behavior of a class of genetic adaptive systems, Ph.D. dissertation, Univ. of Mich., Ann Arbor, 1975.

Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley-Longman, Reading, Mass., 1989.

Goldberg, D. E., and K. Deb, A comparative analysis of selection schemes used in genetic algorithms, in *Foundations of Genetic Algorithms*, pp. 69–93, Morgan Kaufmann, San Francisco, Calif., 1991.

Harik, G. R., E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, The gambler's ruin problem, genetic algorithms, and the sizing of populations, in *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*, pp. 7–12, IEEE Press, Piscataway, N. J., 1997.

Holland, J. H., *Adaptation in Natural and Artificial Systems*, Univ. of Mich. Press, Ann Arbor, 1975.

McKinney, D. C., and M. D. Lin, Genetic algorithm solution of groundwater management models, *Water Resour. Res.*, *30*(6), 1897–1906, 1994.

Reed, P., B. Minsker, and A. J. Valocchi, Cost-effective long-term groundwater monitoring design using a genetic algorithm and global mass interpolation, *Water Resour. Res.*, this issue.

Ritzel, B. J., J. W. Eheart, and S. Ranjithan, Using genetic algorithms to solve a multiple objective groundwater pollution containment problem, *Water Resour. Res.*, *30*(5), 1589–1603, 1994.

Thierens, D., Analysis and design of genetic algorithms, doctoral dissertation, Kathol. Univ. Leuven, Leuven, Belgium, 1995.

Thierens, D., and D. E. Goldberg, Mixing in genetic algorithms, in *Proceedings of the Fifth International Conference on Genetic Algorithms*, edited by S. Forrest, pp. 38–45, Morgan Kaufmann, San Francisco, Calif., 1993.

Thierens, D., and D. E. Goldberg, Convergence models of genetic algorithm selection schemes, in *Proceedings of the Third Conference on Parallel Problem Solving From Nature*, edited by Y. Davidor, H.-P. Schwefel, and R. Manner, pp. 119–129, Springer-Verlag, New York, 1994.

Thierens, D., D. E. Goldberg, and A. Guimaraes Pereira, Domino convergence, drift, and the temporal-salience structure of problems, in *The 1998 IEEE International Conference on Evolutionary Computation Proceedings*, pp. 535–540, IEEE Press, New York, 1998.

Wang, Q. J., The genetic algorithm and its application to calibration conceptual rainfall-runoff models, *Water Resour. Res.*, *27*(9), 2467–2471, 1991.

D. E. Goldberg, Department of General Engineering, University of Illinois, 117 Transportation, MC-238, 104 S. Matthews Avenue, Urbana, IL 61801. (deg@uiuc.edu)

B. Minsker and P. Reed, Department of Civil and Environmental Engineering, University of Illinois, 3230 NCEL, MC-250, 205 N. Matthews Avenue, Urbana, IL 61801. (minsker@uiuc.edu; preed@uiuc.edu)

(Received January 18, 2000; revised July 3, 2000;
accepted July 31, 2000.)