The Practitioner's Role in Competent Search and Optimization Using Genetic Algorithms

Patrick M. Reed*, Barbara S. Minsker*, and David E. Goldberg**

*3230 Newmark Civil Engineering Laboratory, Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, 205 N. Mathews Ave., Urbana, IL, 61801, PH (217) 333-6967; FAX (217) 333-6968; email: preed@uiuc.edu, minsker@uiuc.edu

**117 Transportation, Department of General Engineering, University of Illinois at Urbana-Champaign, 104 S. Mathews Ave., Urbana, IL, 61801; email: deg@uiuc.edu

Abstract

In the past decade genetic algorithms (GAs) have been used in a wide array of applications within the water resources field. Although usage of GAs has become widespread, the theoretical work from the genetic and evolutionary computation (GEC) field has been largely ignored. Most practitioners have instead treated the GA as a black box, specifying the parameters that control how the algorithms navigate the spaces of each application using trial-and-error analysis. Trial-and-error analysis is a time-consuming, difficult process resulting in an arbitrary selection of parameters without any regard to the fundamental properties of the GA. The concept of "competent search and optimization" as discussed in this work addresses this difficulty by using the available theoretical work from the GEC field to set the population size, the selection pressure, account for potential disruptions from crossover and mutation, and prevent drift stall. This paper provides an overview of a three-step method for utilizing GEC theory to ensure competent search and avoid common pitfalls in GA applications.

Introduction

GAs have the appeal of being a robust, non-derivative based technique capable of discrete and continuous optimization for single and multiobjective problems. This appeal has resulted in a significant increase in their use over the past decade to solve a wide array of problems within the water resources field. Although the popularity of this solution technique has increased, little focus has been placed on the actual design of the GA to ensure competent search and optimization. Practitioners often cite the difficulty in identifying robust parameter settings that will ensure evolution-based search will identify high quality solutions (see Aly and Peralta 1999). Trial-and-error analysis has been the traditional approach within the field of water resources for setting the control parameters for genetic algorithms. Trial-and-error methods inherently treat genetic algorithms as black boxes and ignore the theoretical work of the GEC field.

The approach discussed in this paper treats the GA as a system with discernable properties and utilizes valuable theoretical work from the GEC field to discern these properties. This paper is meant to be a illustrative companion to the rigorous GA design methodologies for computationally intensive single and multiple objective applications given in Reed et al. (2000) and Reed et al. (2001), respectively. This paper assumes a working knowledge of the GA and its terminology (see Goldberg 1989 for an introduction).

Competent Search and Optimization: From Theory to Practice

Goldberg (1998) defines *competent GAs* as "…GAs that solve hard problems quickly, reliably, and accurately—through a combination of effective (1) *design methodology*, (2) *design theory*, and (3) *design*." Goldberg (1998) also describes the *control map* concept for GAs where theoretical relationships for population sizing, selection pressure, crossover, and mutation are used to find values for the control parameters that identify a performance "sweet spot" where competent search and optimization is enabled. The *design theory* for the competent GA and the concept of the control map for GA performance are largely derived from Goldberg et al. (1992) and Thierens (1995), respectively. Although these studies provided a theoretical basis for the design of GAs, practitioners still faced the difficulty of using relationships that require the specification of parameters that are not readily identifiable in practice such as building block (BB) size and order. BB's blocks are highly relevant subsets of the binary digits representing designs and are used by the GA to construct optimal solutions.

This difficulty was addressed by the GA *design methodology* presented by Reed et al. (2000) for single objective applications, which was extended for multiobjective applications by Reed et al. (2001). The design methodology uses the theoretical framework of Thierens (1995) to develop a simple 3-step approach for setting the GA's parameters that ensures convergence to high quality solutions (i.e. the "sweet spot"). The methodology progresses through the following three steps: (1) preliminary problem analysis, in which the problem's formulation is used to obtain estimates for population sizing and the computational complexity of using a GA to solve the application, (2) parameter selection, in which theoretical relationships are used to set the probabilities of crossover and mutation (as well as for sizing niches if required), and (3) elitism and drift analysis are performed using a minimum number of trial runs to ensure that the population size is sufficiently large. The works of Thierens (1995) and Reed et al. (2000, 2001) provide a *design theory* and a *design methodology* satisfying only two of the three requirements for the definition of a competent GA. The GA practitioner is the key to satisfying the definition of the competent GA. She or he must utilize these tools in an effective *design* for the GA when solving her or his application. Further discussion on the design methodology in Reed et al. (2000, 2001) is given below.

Three Steps on the Road to the Competent GA

Reed et al. (2000, 2001) provide a 3-step methodology for selecting the proper input parameters and ensuring competent search and optimization. These studies have the intent of stepping beyond the black box mystique implicit to trial-and-error error design methods for GAs, which can often lead to unsubstantiated generalizations about the algorithm's effectiveness. Theoretical relationships allow engineering practitioners to gain direct insights into the algorithm's performance as a function of its input parameters while also reducing the number of parameter combinations that must be considered relative to trial-and-error methods. Minimizing the number of parameter combinations that must considered is vital in computationally intensive applications where a single trial run can take days. The subsequent sections provide an illustrative overview of the important concepts and considerations implicit to each of the 3 design steps.

Initial Considerations. The design methodology assumes that computationally intensive fitness functions for water resources applications preclude identifying parameter settings for a distribution of

initial random number seeds and instead focuses on finding optimal parameter settings for a single random number seed. Additionally, the method assumes that the practitioner has successfully formulated his or her problem such that the GA will converge to a feasible solution (or a feasible nondominated set in the case of a multiobjective application).

Step 1: Preliminary Problem Analysis. The first step in designing a competent GA consists of performing some basic calculations to determine a range of population sizes and the computational complexity of solving the problem as it is currently formulated. The computational complexity associated with solving an application can be estimated by multiplying the population size by the number of generations required for the selection scheme being used to converge. Both population sizing and convergence rates are discussed below.

Population Sizing. Reed et al. (2000) and Reed et al. (2001) use population-sizing relationships from Harik et al. (1997) and Mahfoud (1995), respectively. These relationships require the practitioner to define how the population size grows as a function of the application's BB order. The size and order of BBs are not readily identified in real-world applications and therefore must be conservatively estimated. Both Reed et al. (2000) and Reed et al. (2001) assume that BBs will have an order less than or equal to 5. This assumption is based on the Schema Theorem, which states that only short, low order BBs will survive the operators of selection, crossover, and mutation. Applying this assumption yields a range of five discrete potential population sizes for the practitioner to consider. An important parameter to consider when estimating the range of population sizes in applications where constraints are enforced using penalty functions is the standard deviation of the population fitness.

The fitness standard deviation is directly proportional to the population size required to solve an application and should therefore be kept to a minimal value. Penalty weights often control the magnitude of the fitness standard deviation, therefore excessive penalties for constraint violations will cause population size estimates to be unreasonably large. It is important for the practitioner to see the direct relationship between her or his problem formulation and the ability of the GA to identify optimal solutions. If unreasonably large population size estimates are attained, the practitioner must reduce the magnitude of the penalty weights while also ensuring that the GA will seek and converge to feasible solutions. It must be understood that penalty functions severely complicate the decision space of an application and will require an iterative approach to problem formulation to ensure that both the fitness standard deviation is sufficiently small and feasible solutions are being sought. Michalewicz & Schoenauer (1996) and Hilton & Culver (2000) provide a review and guidance for constraint handling methods for evolutionary computation applications. Proper constraint handling methods can reduce the overall computational complexity of solving an application by both reducing the required population size and improving the GA's convergence rate. Additionally, practitioners may want to consider reformulating constrained, single objective problems to be unconstrained, multiobjective applications when possible (for a discussion of this topic see Fonseca & Fleming 1998).

Convergence Rates. Convergence rate relationships for other available selection schemes can be referenced in Thierens et al. (1994, 1998). The convergence rate of a GA is dependent on the selection scheme used to propagate highly fit individuals. In single objective applications, tournament selection has been the preferred selection scheme because Goldberg and Deb (1991) showed that the method is robust and less prone to premature convergence. In multiobjective applications, stochastic remainder selection is most frequently used for rapid convergence rates because fitness sharing maintains population diversity and prevents premature convergence (Fonseca & Fleming 1995, Coello 1999, Van Veldhuizen 1999). Tournament selection and stochastic remainder selection are assumed in

Reed et al. (2000) and Reed et al. (2001), respectively, when estimating convergence rates. It is important when designing GAs to ensure that the algorithm converges. Failure to converge in timely manner can reflect excessive penalties in the problem formulation or insufficient selection pressure, which in either case will result in the sub-optimal performance for the GA. Selection pressure represents the GA's ability to ensure that only highly fit individuals are allowed to pass their traits to latter generations.

Step 2: Parameter Selection. Step 1 provides a range of population sizes and an estimate of run length. The remaining input parameters are specified in this step. For single objective applications, Reed et al. (2000) present a relationship from Thierens (1995) for an upper boundary on the probability of crossover P_c. The upper bound relationship provides a disruption boundary for crossover that varies as a function of selection pressure and allows the user to reduce the potential disruptive effects of the recombination operator. The boundary stipulates that the practitioner must increase tournament sizes beyond the traditional binary tournament if values of P_c above 0.5 are used. For multiobjective applications, Reed et al. (2001) present a range of crossover probabilities that have been successful in multiobjective applications (0.6 to 0.9) and associated population-sizing relationships from Mahfoud (1995) that directly account for the potential disruptive effects of crossover.

Goldberg (1998) discusses the relationship between selection and crossover in terms of a *race* between innovation and convergence. Selection exerts a takeover force on the GA that causes the algorithm to converge to a single solution (or set of solutions in multiobjective applications). Crossover exerts an opposing innovative force that enables the GA to explore new regions of the decision space as new designs emerge from recombination. The *race* requires that the time scale for innovation must be shorter than the time scale of convergence in order to prevent premature convergence. Reed et al. (2000,2001) recommend setting the value of P_c as high as possible while still satisfying the disruption performance boundary given by Thierens (1995) for single objective applications and maintaining feasible population size estimates for multiobjective applications.

The above relationships assume that the mutation operator is minimally disruptive. This assumption is enforced in Reed et al. (2000, 2001) by setting the probability of mutation P_m equal to the inverse of the population size as recommended by DeJong (1975). Multiobjective applications require the additional specification of the parameters controlling fitness sharing. Reed et al. (2001) give the relationships for setting these parameters from Deb & Goldberg (1989). Step 3 further explores the role of selection pressure and uses a limited number of trial runs to select an optimal population size from the range computed in step 1.

Step 3: Elitism & Drift Analysis. Domino convergence occurs when different subsections of the binary strings composing a population converge at variable rates as a function of their relevance to the final solution (see Thierens et al. 1998). The common occurrence of domino convergence in real-world applications requires further analysis of the role of selection in the performance and design of the GA. When some portions of the binary strings (or decision variables) are more important to the final solution than others, *genetic drift* can occur. Genetic drift stall occurs when the portions of the binary string representing decision variables that have a reduced "salience" to the final solution do not experience sufficient selection pressure and converge to non-optimal values under the random fluctuations associated with crossover and mutation (Thierens et al. 1998).

It is very important for a practitioner to properly set the appropriate selection pressure for his or her application to avoid drift stall. Figure 1 gives an example plot of an improperly designed GA experiencing drift stall.

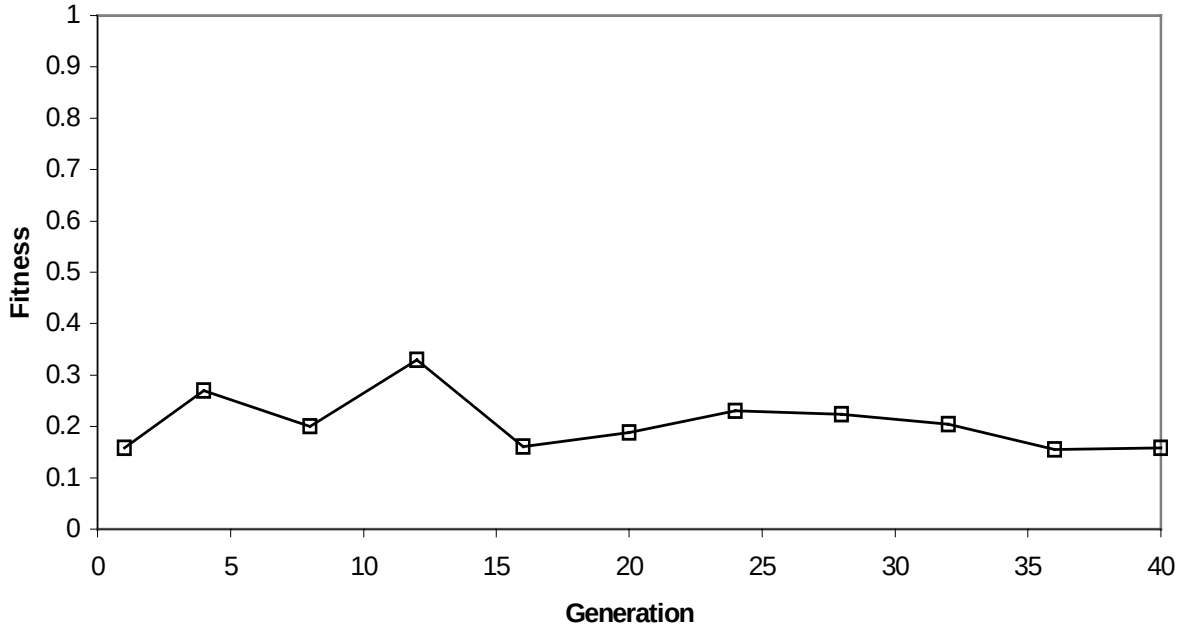

Figure 1. Illustration of genetic drift stall

Drift stall can be readily identified when the fitness of the best individual (for a single objective) or a scoring metric representing the quality of a nondominated front (for multiple objectives) fails to monotonically increase over the duration of a run. The GA's performance as a function of time (or generation) can be measured using either *online performance analysis* or *offline performance analysis*. Online performance analysis tracks the current best solution (or the quality of the nondominated front) in a single generation without considering previous generations. Alternatively, offline performance analysis keeps track of the best individual (or the collection of nondominated individuals) from all generations preceding and including the current generation. Both performance measures are equally valid and can be used based on the practitioner's preference. It should be noted that online performance analysis is the more stringent measure.

Plots of GAs experiencing drift should not be used to criticize the solution method in empiric performance comparisons with other methods unless the practitioner presents proof that the algorithm has been carefully designed for competency and explicitly states which performance measure she or he is using. Generalizations on algorithmic performance are not valid when the GA is treated as a black box and an arbitrary selection of input parameters are used. In many cases, drift stall reflects the practitioner's failure to properly set the selection pressure through increasing the tournament size, the population size, or through some form of elitism (Reed et al. 2000, 2001).

Elitism. Elitist operators provide a means of reducing genetic drift by ensuring that the best individual(s) are identified and allowed to pass their traits to latter generations. Elitism serves to rescale the system such that low "salience" portions of the binary strings in the population are no

longer lost due to deficient selection pressure (Thierens et al. 1998). This concept is straightforward in single objective optimization, where a single individual with the current best objective function value is inserted into the next generation. In multiobjective applications some fraction of the solutions along the current nondominated front must be passed on to the next generation (see Reed et al. 2001, Zitzler & Thiele 1999, Parks & Miller 1998, Bäck 1996, and Ishibuchi & Murata 1996 for a description of alternative multiobjective elitist strategies). Elitism has been shown to improve the performance and convergence of the GA in both single and multiple objective applications (Thierens et al. 1998, Zitzler et al. 2000, Reed et al. 2001).

Population sizing using trial runs. The final step in designing a GA requires that a single population size be selected from the five population sizes computed in step 1 of the design methodology. Reed et al. (2000, 2001) recommend a rule-of-thumb where a minimum number of successive runs are performed for increasing population sizes, starting with the smallest population size, until the GA's performance does not appreciably increase. The trial runs for increasing population sizes serve two purposes. First, the method increases population sizes until any potential performance problems from random seed selection are no longer an issue. This ensures that robust parameter settings are identified for any specified random seed. The second benefit of successively increasing the population size is the reduction in the potential for drift stall. Thierens et al. (1998) showed that the time scale for genetic drift to occur is a function of population size. Detailed performance analysis of the design methodology can be referenced in Reed et al. (2000) and Reed et al. (2001).

Conclusions

The competent GA requires consideration of the theoretical relationships describing the algorithm's fundamental properties in a design methodology that is accessible to real-world practitioners. This paper summarizes such a methodology, which is given in detail in Reed et al. (2000, 2001) for the simple GA and multiobjective GA, respectively. The most important component in the definition of the competent GA is the role of the practitioner. To ensure good performance, practitioners must use competent design principles in their applications. The competent GA is more concept than computer algorithm in that capricious boundaries are not set to define the exact form of selection, crossover, and mutation that practitioner must use. The competent GA conceptual approach avoids black box trial-and-error analysis and instead makes the practitioner aware of the tradeoffs, limitations, and properties of the selecto-recombinative operators specific to his or her application. Competency enables valuable empirical comparison of GEC methods that are designed for optimal or near optimal performance with other search and optimization techniques. The utilization of current and future theoretical relationships from the GEC field will reduce unsupported generalizations both positive and negative on the effectiveness of evolution-based solution techniques in water resources applications.

References

Aly, Alaa, H. and Richard C. Peralta (1999) Comparison of a genetic algorithm and mathematical programming to the design of groundwater cleanup systems, *Water Resources Research*, 35(8), 2415-2425.

Bäck, T. (1996) *Evolutionary Algorithms in Theory and Practice*, Oxford University Press: New York, NY.

Coello, C.A.C. (1996) *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design*, Doctoral dissertation, Tulane University: New Orleans, LA.

Coello, C.A.C (1999) A comprehensive survey of evolutionary-based multiobjective optimization techniques, *Knowledge and Information Systems*, 1(3), 269-308.

Deb, K. & Goldberg, D. E. (1989) An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer, ed. *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 42-50, Morgan Kaufmann: San Mateo, CA.

DeJong, K. A. (1975) *An analysis of the behavior of a class of genetic adaptive systems*, Doctoral dissertation, University of Michigan: Ann Arbor, MI.

Fonseca, C. M. and Fleming, P. J. (1995) An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1), 1-16.

Fonseca, C. M. and Fleming, P. J. (1998) Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms—Part I: A Unified Formulation, *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 28(1):26-37.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley: New York, NY.

Goldberg, D. E., and K. Deb, (1991) A comparative analysis of selection schemes used in genetic algorithms, in *Foundations of Genetic Algorithms*, pp. 69-93, Morgan Kaufman, San Mateo, CA.

Goldberg, D. E., K. Deb, and J. H. Clark (1992) Genetic algorithms, noise, and the sizing of populations, *Complex Systems*, 6, 333-362.

Goldberg, D. E. (1998) *The Race, the Hurdle, and the Sweet Spot: Lessons from Genetic Algorithms for the Automation of Design Innovation and Creativity*, Department of General Engineering, University of Illinois at Urbana-Champaign, ILLIGAL Report No. 98007.

Harik, G.R., E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. (1997) 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, New York, NY.

Hilton, Amy Chan B. and Culver, T. (2000) Constraint Handling for Genetic Algorithms in Optimal Remediation Design, *J. Water Resources Planning and Management*, 126(3): 128-137.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan: Ann Arbor, MI.

Ishibuchi, H. & Murata, T. (1996) Multi-objective genetic local search algorithm. In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pp. 119-124, IEEE: Piscataway, NJ.

Mahfoud, S. (1995) *Niching Methods for Genetic Algorithms*, Doctoral dissertation, University of Illinois: Urbana, IL.

Michalewicz, Z. & Schoenauer, M. (1996) Evolutionary Algorithms for Constrained Parameter Optimization Problems, *Evolutionary Computation*, 4(1): 1-32.

Parks, G. T. & Miller, I. (1998) Selective breeding in a multiobjective genetic algorithm. In A. E. Eiben, T. Bäck, M. Schoenauer, and H. Schwefel, eds. *Fifth International Conference on Parallel Problem Solving form Nature*, pp. 250-259, Springer: Berlin, Germany.

Reed, P., Minsker B., & Goldberg, D. E. (2000) Designing a competent simple genetic algorithm for search and optimization. *Water Resources Research*, 36(12), 3757-3761.

Reed, P., Minsker B., & Goldberg, D.E. (2001) A multiobjective approach to cost effective long-term groundwater monitoring using an Elitist Nondominated Sorted Genetic Algorithm with historical data, *Journal of Hydroinformatics*, in press.

Srinivas, N. & Deb, K. (1995) Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3), 221-248.

Thierens, D. & Goldberg, D. E. (1993) Mixing in genetic algorithms. In S. Forrest, ed. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 38-45, Morgan Kaufmann Publishers: San Mateo, CA.

Thierens, D. & Goldberg, D. E. (1994) Convergence Models of Genetic Algorithm Selection Schemes, In Y. Davidor, Hans-Paul Schwefel, and Reinhard Manner, eds. *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, pp. 119-129, Springer-Verlag: New York, NY.

Thierens, D., (1995) *Analysis and design of genetic algorithms*, Doctoral dissertation, Katholieke Universiteit Leuven, Leuven, Belgium.

Thierens, D., Goldberg, D. E., & Pereira, A. G. (1998) 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, NY.

Van Veldhuizen, D. A. (1999) *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, Doctoral dissertation, AFIT/DS/ENG/99-01, Air Force Institute of Technology: Wright-Patterson AFB, Ohio.

Zitzler, E. & Thiele, L. (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), 257-271.

Zitzler, E., Deb, K., & Thiele, L. (2000) Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation*, 8(2), 173-195.