A hydroinformatician's approach to computational innovation and the design of genetic algorithms*

David E. Goldberg

ABSTRACT

The paper discusses how a computational hydralician got involved in the design, implementation, and application of *genetic algorithms* – search procedures based on the mechanics of natural selection and genetics – and how that involvement depended critically upon the modern hydroinformatician's sense of appropriate modeling of complex phenomena such as fluid mechanics. The paper starts by briefly reviewing the mechanics of genetic algorithms and then connects that mechanics to the *fundamental intuition* that GAs have something in common with human innovative processes. The paper continues with a short aside on a difference in the way hydroinformaticians and computer scientists are taught to reason with models. This leads to a discussion of the race between selection and recombination in a GA, and how understanding the race leads immediately to the construction of a critical dimensionless quantity in GA analysis. This dimensionless quantity is then sketched in the GA's *control map*, and the paper concludes with a brief discussion of how such knowledge leads to the design of *competent GAs* – GAs that solve hard problems, quickly, reliably, and accurately. The paper concludes with an invitation to hydroinformaticians to both use genetic algorithms in the solution of difficult hydroinformatics problems and to apply their analytical skill to the design of ever more capable genetic and evolutionary algorithms.

Key words | fluid mechanics, genetic algorithms, hydraulics, hydroinformatics, modeling

David E. Goldberg
Department of General Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801,
USA
E-mail: *deg@uiuc.edu*
http://www-illigal.ge.uiuc.edu/

INTRODUCTION

I was pleased by the invitation to write an article for the *Journal of Hydroinformatics* for three reasons, one intellectual, one personal, and one speculative. The intellectual reason is that I value the opportunity to share some recent results on the design of effective *genetic algorithms* (GAs) – computer search procedures based on the mechanics of natural selection and genetics. The personal reason is that I started my academic career as a computational hydraulician – someone who would today be pleased to call himself a hydroinformatician – and I welcomed the opportunity to discuss how this background and intellectual outlook has shaped the way I analyze and design GAs. My GA friends know that John H. Holland, the inventor of genetic algorithms, was one of my PhD advisors at the University of Michigan, but they are largely unaware of the influence on my thinking by my other PhD advisor, E. Benjamin Wylie, the noted fluid transients authority. Here I pay homage to Ben and the subtle lessons he taught me about the rich complexity of the hydro-world, and how those lessons have led to a design methodology for genetic algorithms that works. Finally, my speculative reason has to do with wanting to present a hypothesis that has gnawed at me for the better part of 20 years: that is, that genetic algorithms have something in common with what we call *human innovation*.

So these are the reasons I was pleased to be asked to write the paper, but to try to present a coherent story, let's proceed in the following way. First, let's briefly review what genetic algorithms are, and then let's make our initial pass at understanding the connection between GAs and

*Portions of this paper have been excerpted from a paper by the author entitled '3 lessons of genetic algorithms for computational innovation', *Hydroinformatics '98*, 1–7.

innovation at an intuitive level. Then, let's digress briefly and discuss a curious difference between we hydroinformaticians on the one hand and computer scientists on the other. This apparent digression will set the stage for my hydraulician's approach to modeling and designing complex conceptual objects like genetic algorithms. From there we will briefly review some key GA design theory that borrows from one of the hydraulician's favorite tools, dimensional analysis, and then we will look at how that theory has helped create *competent* genetic algorithms – GAs that solve hard problems, quickly, reliably, and accurately – and how competent GAs are a first-order computational model of the processes of human innovation.

THE NICKEL TOUR OF GENETIC ALGORITHMS

Elsewhere, I have written at length (Goldberg 1989) about GA basics, and here we briefly review the elements of GA mechanics.

GAs search among *populations of chromosomes* that may be decoded to represent *solutions* to some *problem*. Basic to the operation of a GA is the ability to evaluate the *fitness to purpose* of one solution relative to another. With a population in place, *selection* and *genetic operators* can process the population iteratively to create a sequence of populations that hopefully will contain more and more good solutions to our problem as time goes on. There is much variety in the types of operators that are used in GAs, but quite often (1) *selection*, (2) *recombination*, and (3) *mutation* are used. Simply stated, selection allocates greater survival to better individuals, recombination combines bits and pieces of parental solutions to form, new, possibly better offspring, and mutation makes one or more changes to an individual's trait or traits.

THE FUNDAMENTAL INTUITION

Our nickel tour doesn't do justice to GA mechanics, but even a longer explanation would leave us scratching our heads and asking how such simple operators might do anything useful, let alone promote an effective, robust search for good stuff. It is something of an intellectual mystery to explain why such individually uninteresting mechanisms acting in concert might together do something useful. Starting in 1983 (Goldberg 1983), I have developed what I call the *fundamental intuition of genetic algorithms* or the *innovation intuition* to explain this apparent mystery. Specifically, I liken the processing of selection and mutation taken together, and that of selection and recombination taken together, to *different facets of human innovation*, what I will call the *improvement* and *cross-fertilizing* types of innovation. Here we will concentrate on the cross-fertilizing type of innovation exclusively.

Selection+recombination=innovation

To understand how selection and crossover might give us anything useful, we appeal to our own sense of human cross-fertilizing innovation. What is it that people do when they are being innovative in a cross-fertilizing sense? Usually they are grasping at a notion – a set of good solution features – in one context, and a notion in another context and juxtaposing them, thereby speculating that the combination might be better than either notion taken individually. My first thoughts on the subject were introspective ones, but others have written along similar veins, for example, the French poet-philosopher Valéry:

> It takes two to invent anything. The one makes up combinations; the other chooses, recognizes what he wishes and what is important to him in the mass of the things which the former has imparted to him.

Verbal descriptions are far from the exacting rigor of computer code, but it is interesting that the innovation intuition has been articulated by philosophers of earlier times. In a moment, we will see if we can't make this connection between innovation and genetic algorithms a little tighter with some quantitative theory, but right now we pause to explore a curious difference between hydroinformaticians and computer scientists.

A difference between hydroinformaticians and computer scientists

As I've tried to better understand the operation of genetic algorithms, I've approached the subject with the toolkit

that my training in computational hydraulics provided me. Over the years, it has seemed to me that my experience with the toolkit has been one of those ubiquitous good news, bad news stories. The good news has been that the analytical toolkit of my training has permitted both the successful analysis and design of increasingly complex and effective GAs. The bad news has been that I've come to realize how alien that type of theory is to the mass of GA practitioners and theoreticians who largely come to the subject through training in modern computer science. As I've struggled to understand this situation, I've come to realize that there are biases in the education of these two groups, and that those biases fundamentally shape how they approach the analysis of problems.

A bias in the education of a fluids engineer exposed

There is a bias in the training of hydroinformaticians, hydraulicians, and for that matter, all engineers that largely goes unremarked, but it subtly shapes the way such people approach the analysis of problems. Specifically, engineers of all stripes are taught *to apply the simplest model that will enable the effective design of the object system or technology*. This sounds fairly obvious, and to engineers it is almost second nature, but the bias is pervasive in the engineering curriculum. For example, engineers are taught statics before dynamics. They are taught linear systems before nonlinear ones. Ideal flow is taught before the Navier–Stokes equations. Strength of materials is taught before elasticity, and so on. In short, engineers are taught a number of different models for the same phenomenon, and their ordering in the curriculum implicitly teaches the engineer to grab for the simplest model first in the hopes that it will be sufficient to solve the design issue at hand.

A bias in the education of computer scientists exposed

The engineer's training contrasts rather sharply with the treatment of analysis in the education of scientists, particularly computer scientists. In physics, Newton's second law is taught first, and equilibrium is taught as a rather uninteresting special case. In computer science, study of theory begins with theorems and proofs in discrete mathematics, and respect for rigor in derivation is valued very strongly. CS theoreticians are taught to advance the *state of theory* by the development of ever more exact calculations for different algorithms.

Note that the perspective of the engineer and that of the scientist are both reasonable ones, *depending upon the goals of the study*. If one is designing a complex system, the relaxation of rigor for applicable design theory could not be more useful. If one is attempting to develop ever more rigorous models, than theorem-proof is the way to go. The problem comes when individuals become confused as to their goals. If one is attempting to design better systems – whether they be airplanes, toasters, or genetic algorithms – it is puzzling why a designer would reflexively adopt the rigor-above-all stance of the CS theoretician. But the firewall that exists between theory as taught in CS and theory as needed in practice makes it difficult for computer scientists to grab for the right tool at the right time.†

Stated somewhat differently, if one's goals are the development of more rigorous models, then by all means be more rigorous. But if one's goals are the design of some complex system such as a genetic algorithm, then theory is useful as it is *instrumental* toward the design of better systems. Elsewhere (Goldberg 1996), I have put this argument on firmer footing through a discussion of *economic models*, where I use an analysis of the *costs* and *benefits* of modeling toward the advancement of some technology. We will not pursue that analysis here, but instead we will examine one economic model in detail by deriving a scaling law for innovation on dimensional grounds.

A DIMENSIONAL MODEL OF INNOVATION: TWO TIME SCALES AND THE RACE

Our digression may puzzle some, and indeed when I present or write for engineering audiences particularly those familiar with the importance of dimensional arguments in the study of the hydro-world, I am preaching to

†To keep the argument symmetrical, it is also hard for the trained engineer to put on the rigor cap when his goals shift toward the development of pure theory.

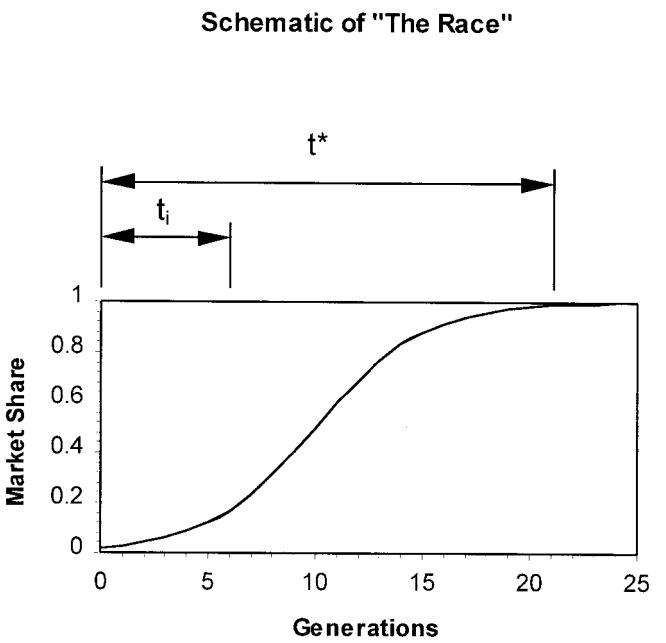

Figure 1 | A schematic of the race illustrates the tension between between innovation and selection. When the innovation time t_i is less than the selection takeover time t*, innovation proceeds apace (steady-state innovation). When the situation is reversed, innovation is stalled (premature convergence).

the choir. Nonetheless, it is important to understand how the arguments to be made in this section may be controversial to some, even if they seem eminently reasonable to us. Returning to our discussion of the innovation intuition – the assertion that the process of selection and exchange of genetic algorithms is similar to a kind of innovation displayed by human beings – it would be nice to go beyond mere hand waving and understand some of the underlying issues of innovation. The first key to greater insight is to understand the critical *race* that goes on in a competitive innovating system. In an evolving system acted upon by selection alone, we expect to see an S-shaped time history of the market share of the best individuals in the population (Figure 1), and we may calculate the time it takes to go from a small market share to a large one as a characteristic *takeover time* or t^*. This seems reasonable enough, but real GAs have selection and recombination. What difference could it possibly make to understand the behavior of a competitive system under selection alone?

The answer to this question comes quickly and convincingly if we *imagine another characteristic time*, call it the *innovation time* t_i, which we shall define as the mean time for recombination or innovation operator to achieve a solution better than any achieved to this point. With such a characteristic time in mind there are two basic situations that we must be concerned with: the situation where the takeover time is greater than or equal to the innovation time, $t^* \geq t_i$, and that where the innovation time exceeds the takeover time $t^* < t_i$.

In thinking about these two situations, we immediately wonder which is the more advantageous for a selecto-recombinative GA, and the answer is apparent with some straightforward reasoning, as follows. The condition where innovation time leads (is less than or equal to) the takeover time is most advantageous for continuing innovation, because prior to the best individual dominating the population, recombination *creates a better individual*. Thereafter this better individual starts to dominate the population, and in essence, the innovation clock is reset. This cycle of partial takeover and continued innovation is repeated over and over again, resulting in the happy condition I have dubbed *steady-state innovation*.

Contrast this virtuous setting with the condition where innovation time lags (is greater than) takeover time. In such a situation, the current best guy continually increases in market share without serious competition and ultimately takes the population to substantial convergence, and now it is too late because *diversity is a necessary condition of selectorecombinative success*. This situation was called premature convergence (De Jong 1975) fairly early in the GA literature, but until the introduction of the above *time scales* argument (Goldberg *et al*. 1993*b*), there was no means of analyzing the boundary between innovative success and failure. With the understanding of the crucial role of time scales and the race, rational analysis and design of competitive innovating GAs has advanced quite rapidly.

A GA'S CONTROL MAP AND ITS SWEET SPOT

One of the tools critical to these rapid advances is the so-called *control map*, which helps us delimit a genetic

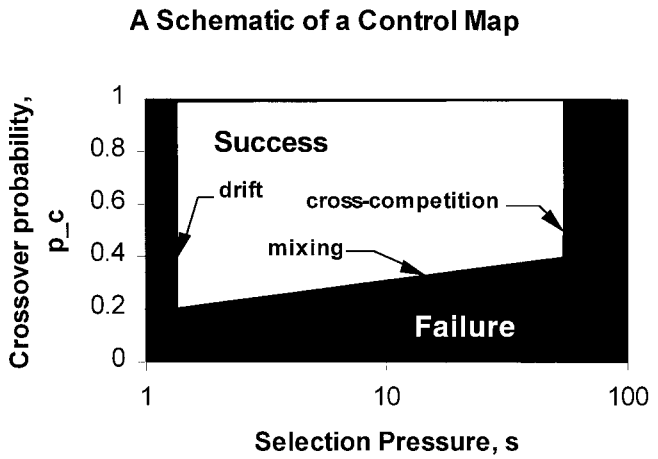

Figure 2 | A control map shows the GA's sweet spot or zone of success for setting GA control parameters. The mixing boundary grows as the logarithm of the selection pressure s, as may be derived from the dimensional argument of the race presented in this paper.

algorithm's *sweet spot*. Technical details of these developments are in the original papers (Goldberg *et al*. 1993*b*; Thierens 1995; Thierens & Goldberg 1993), but here we strive for qualitative understanding of the key points. These can best be obtained by focusing on the schematic of the sweet spot of a genetic algorithm operating on an easy problem as shown in Figure 2. In this map, we plot the feasible settings of the GA's control parameters, *s*, the selection pressure, and p_c, the probability of crossover. The selection pressure is simply the number of copies that are given to the best individual in the population under selection alone. The crossover probability is the frequency with which mated chromosomes actually undergo the exchange of crossover.

In the previous section, we discussed the race between innovation and selection. If we take this argument seriously and develop an equation from the condition when the takeover time is of the same order as the innovation time, we obtain the mixing or innovation boundary shown on the graph, where the crossover probability must increase as the logarithm of the selection pressure. Any value of crossover probability above this line is expected to succeed and any value below this boundary is expected to fail.

There are two other boundaries shown on the control map. To the left we see the region of success bounded by the so-called *drift boundary*, where convergence is controlled by the vagaries of random fluctuation when the selection pressure is small. To the right, we see the region of failure dominated by what we call *cross-competition*, when semantically independent traits end up competing with one another when the selection pressure is close to the population size.

THE HURDLE TO EFFECTIVE MIXING

GA control maps and the sweet spot can be used with some precision to predict the success (or failure) of a given GA on some specified problem. It has been demonstrated both theoretically and empirically elsewhere (Goldberg *et al*. 1993*b*) that easy problems – problems that may be solved through bitwise exchanges – have large sweet spots, and almost any selectorecombinative GA with any reasonable choice of crossover operator can be expected to do well in such cases. On the other hand, as a problem becomes more difficult – that is, as a problem has building blocks larger than single bits – the size of the sweet spot shrinks even as the population size is increased nominally to account for the increased noise of the more difficult problem instance. This is a big problem, and ultimately the sweet spot vanishes. Another way to view the same problem is to ask the question, what size population is required to solve problems of increasing difficulty and length. Both theoretically and empirically it has been shown (Thierens & Goldberg 1993; Thierens 1995) that population sizes must grow exponentially to accommodate harder problems.

This leaves us with a split decision regarding the efficacy of simple GAs. If a problem is bitwise solvable, modest population sizes may be used and accurate, reliable solutions may be expected in small numbers of function evaluations, and we should expect those numbers to scale well, growing no more quickly than a subquadratic function of the number of decision variables or bits. On the other hand, with a more difficult problem instance, simple recombination operators scale badly, requiring a

superexponential number of function evaluations to get reliable answers to even boundedly difficult problems, and therein lies the rub, the hurdle to the design of competent GAs. Is there some way to design crossover mechanisms that allow solutions to hard problems to scale more like those of easy problems?

COMPETENT GA DESIGN: 1993 TO PRESENT

Remarkably, the answer to this question is a resounding 'yes', and the trick is to both *identify* and *exchange* clusters of genes appropriate to solving a problem without the need for human intervention or advice.

A line of work dating back to 1989 (Goldberg *et al*. 1989) set this as its goal and succeeded in achieving this goal for the first time in 1993 (Goldberg *et al*. 1993*a*) with the creation of the *fast messy genetic algorithm* (fmGA). Detailed description of the fast messy GA is beyond the scope of this treatment, but the key to competent GA design is to tame the race between selection and crossover by *identifying building blocks before deciding among them*. This is easy to say, but hard to do. Most naive efforts at accomplishing this goal fail, and subtle designs are necessary to get a competent GA to fire on all cylinders.

Figure 3 shows the results from the first competent GA, the fast messy GA (Goldberg *et al*. 1993*a*). In a problem with order-5 difficult building blocks, the fast messy GA is able to find global solutions in times that grow as a subquadratic function of the number of decision variables, as expected. In contrast, the original messy GA and mutation-oriented hillclimbing (Mühlenbein 1992) have numbers of function evaluations that grow as a quintic function of the number of decision variables, and the exponent on this polynomial gets worse as problems get harder. On the other hand, a competent GA appears to require only subquadratic growth on all problems of fixed difficulty, and this kind of performance characteristic is the kind of robust solution genetic algorithmists have been seeking for so long.

Since 1993, many different competent genetic algorithms have been discovered, including the gene expression messy GA or gemGA (Kargupta 1996), the linkage learning GA (Harik 1997; Harik & Goldberg 1997), and the Bayesian optimization algorithm (Pelikan *et al*. 1999), to name a few. The reader should consult the original literature for more details about these procedures, but a careful comparison would reveal a marked diversity of mechanism. Closer examination, however, would show that each algorithm obeys essentially the same physics. This observation lends support to the suggestion that the kind of theory discussed herein is uncovering important truths that are useful to both the development of advanced genetic algorithms and to a more mechanistic understanding of human innovation.

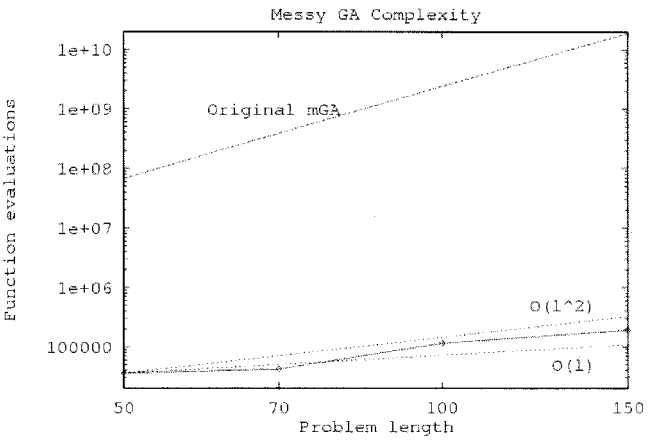

Figure 3 | The fast messy GA results reported in Goldberg *et al*. (1993) demonstrate subquadratic growth in the computation time compared to the original messy GA and simple mutation-based hillclimbers, which grow as the fifth power of the number of variables and are much slower across the board than the fmGA.

INNOVATION AND GAS REDUX

Earlier in the paper I argued that the selection and recombination were possibly powerful search mechanisms by appealing to our intuition of human innovation, but in later sections I argued that GAs done right give us subquadratic, scalable solutions of boundedly difficult problems. This latter possibility is more than a little suggestive that modern competent GA design is giving us something besides very useful practical algorithms to solve hard

problems, quickly, reliably, and accurately. Simply put, it now appears that competent GA design is giving us first-order models of the processes of human innovation. More work needs to be done to connect with both theoretical and experimental results in cognitive science and psychology to see if these computational models hold up under closer scrutiny; however, the very possibility that good engineering design (using the approach of a hydroinformatician, no less) might result in both practical technology and novel science is an interesting one to say the least. At a minimum, the case is a plausible one, and work is ongoing to pursue it further.

CONCLUSIONS AND AN INVITATION

This paper has briefly reviewed genetic algorithms and the fundamental intuition that there may be a connection between the operation of GAs and human innovation. A brief digression explored a difference between hydroinformaticians and computer scientists in terms of their training in the use of models. The paper went on to develop a dimensional argument regarding the relationship between characteristic time scales of takeover and innovation, arguing that when innovation time is relatively fast with respect to takeover that GAs (and other innovating processes) tend to innovate continually and when innovation time is relatively slow with respect to takeover, there is a tendency for progress to stall or prematurely converge. This observation led to the quantitative mapping of a GA's parameter space in a control map, where a sweet spot of high quality convergence was observed. Finally, the effectiveness of these tools for the design of competent GAs was demonstrated by presenting scalability results on a boundedly difficult problem.

These results connect back to my original intellectual, personal, and speculative reasons for writing this paper. Intellectually, the existence of predictive models that can be used in genetic algorithm design has been a blessing, albeit one that has not been widely realized or embraced. It is the hope of this paper that more GA users and practitioners will embrace the kind of scaling laws and modeling demonstrated herein in their GA usage and design.

Personally, and professionally, it is very interesting that the style of analysis that comes so easily to the hydroinformatician because of his or her training in or familiarity with fluid mechanics is so very useful in the realm of complex, conceptual objects such as genetic algorithms. Fluid flows are physical systems and genetic algorithms are conceptual systems, but in a sense, complexity is complexity, wherever it is found. In that way, it is of little surprise that the toolkit that has been so useful in fluid mechanics for the taming of friction, turbulence, convection, shear flow, separation, boundary layers, and a host of other complex fluid phenomena should be so useful in understanding selection, mixing, drift, building blocks, deception, and the many complex phenomena of evolving, innovating systems.

More speculatively, the paper makes the connection between GAs and human innovation somewhat closer. Indeed, there is more work to do, and studies of GA computations and animal, human, and organizational examples of innovation should add to our knowledge a good bit, but at the very least, the results presented here should make the utterance of genetic algorithms and innovation in the same sentence a plausible speech act.

Finally, I close with a twofold invitation to all hydroinformaticians. First, I urge hydroinformaticians everywhere to become familiar with recent work in competent GAs. Scalable, general-purpose algorithms that solve hard problems quickly, reliably, and accurately are immediately applicable to a host of hydroinformatics problems in optimization, system identification, data mining, and machine learning. Although genetic algorithms have been used in hydroinformatics, the general tendency has been to use first-generation GAs, and these have been shown not to scale well with problem size and difficulty. As more hydroinformaticians come to the GA party, and as larger, more complex problems are tackled, the use of competent GAs will not be an option.

Second, I urge analytically minded hydroinformaticians to consider using their well-honed skills at dimensional reasoning and facetwise modeling for the betterment of GAs themselves. As you can see from this paper, relatively straightforward arguments have resulted

in new theoretical understanding of critically important phenomena in GAs and innovation. Although progress has been made on a number of important fronts, much more exciting work remains to be done. Hydroinformaticians are in the enviable position of having the computer expertise to advance the state of GA art and the habits of mind to productively employ scaling laws and other little models in design. Together, these form a significant competitive advantage that could help place hydroinformaticians at the forefront of a growing research area, with important technological and scientific implications.

ACKNOWLEDGEMENTS

Somewhat belatedly, I'd like to take this opportunity to wish Ben Wylie all the best on the occasion of his retirement from the university. Years ago, I knew how his unfailing politeness, organizational skills, attention to detail, and creative and careful approach to computational hydraulics affected my manner of thinking and conduct (I could probably use a refresher course on the first item in the above list of virtues, but this is no fault of Ben's), but more recently I have come to appreciate his profound influence on the way I approach genetic algorithms. Thanks, Ben. You've meant more than you'll know to a generation or two of hydraulics students at the University of Michigan and to your many readers around the world.

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-94-1-0103, F49620-95-1-0338, F49620-97-1-0050, and F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. Support was also provided by a grant from the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are my own and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, the U.S. Army, or the U.S. Government.

REFERENCES

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

Goldberg, D. E. (1983). *Computer-aided pipeline operation using genetic algorithms and rule learning*. Doctoral dissertation, University of Michigan, Ann Arbor.

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

Goldberg, D. E. (1996). The design of innovating machines: Lessons from genetic algorithms. *Computational Methods in Applied Sciences '96*, 100–104.

Goldberg, D. E., Deb, K., Kargupta, H. & Harik, G. (1993*a*). Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 56–64.

Goldberg, D. E., Deb, K., & Thierens, D. (1993*b*). Toward a better understanding of mixing in genetic algorithms. *J. Soc. Instrument Control Engnrs* **32**(1), 10–16.

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems* **3**(5), 493–530.

Harik, G. (1997). *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. Doctoral dissertation, University of Michigan, Ann Arbor.

Harik, G., & Goldberg, D. E. (1997) Learning linkage. *Foundations of Genetic Algorithms* **4**, 247–262.

Kargupta, H. (1996). *SEARCH, Evolution, and the Gene Expression Messy Genetic Algorithm* (Unclassified Report LA-UR 96-60). Los Alamos, NM: Los Alamos National Laboratory.

Mühlenbein, H. (1992). How genetic algorithms really work: I. Mutation and hillclimbing. *Parallel Problem Solving from Nature*, 15–25.

Pelikan, M., Goldberg, D. E., & Cantu-Paz, E. (1999). BOA: The Bayesian optimization algorithm. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, 525–532.

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

Thierens, D., & Goldberg, D. E. (1993). Mixing in genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 38–45.