The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World

David E. Goldberg
Department of General Engineering

IlliGAL Report No. 98004
February 1998

Illinois Genetic Algorithms Laboratory
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801

The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World

David E. Goldberg
Department of General Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801

1 Introduction

When I first wrote about genetic algorithms back in the early 1980s, I searched for a way to convey what I understood to be the power of *genetic algorithms* (GAs)—search procedures based on the mechanics of natural selection and genetics. At that time, I used the metaphor of *human innovation* to convey an intuition for some of the power of operators such as selection, crossover, and mutation, and for many audiences the metaphor seemed apt and satisfying. Later as I began to understand some of the severe limitations of first-generation GAs, the metaphor again was handy, because it helped me think more clearly about how to design *competent* GAs—GAs that solve hard problems, quickly, reliably, and accurately—thereby overcoming the limitations of the then extant technology, and it seemed to me that the next generation of GAs could indeed lay claim to being "innovative" in a definite computational sense. A logician might find fault in the apparent circularity of my reasoning, but an artist or creative designer might instead relish the interplay between the two perspectives—the interplay between thinking of innovation as a model of what GAs do and thinking of GAs as a model of what innovation is. In any event, as time has progressed, I believe the latter perspective has been winning out; that is, as we do a better job of realizing the dream of competent GAs, we are indeed constructing an effective computational model of some of the processes of innovation.

This is a bold claim—and an important one—and in this brief essay I (1) present a lay version of the direct technical lessons of GA research for practitioners interested in innovation, and then (2) interpret those lessons for the understanding of innovation in the real social or organizational context. And make no mistake, as the current business literature attests, innovation is critically important to our increasingly connected global society that propagates innovations from one corner of the globe to another at the speed of light, but the current business literature's lessons are serverely limited because the modeling is largely qualitative, empirical, and anecdotal. An effective computational model should permit us to sharpen our organization and societal pencils and permit us to better recognize true innovation when we see it, design organizations for improved innovation flow, and create support and enabling systems and technology.

This paper starts by reviewing the elements of a genetic algorithm and considers what I call the *fundamental intuition* of genetic algorithms. It continues with a brief discussion of the technical and real-world lessons of GA research for practitioners interested in organizational innovation.

2 The One Minute Genetic Algorithmist

Elsewhere,[1] I have written at length (Goldberg, 1989) about GA basics, and in this section we quickly review the fundamental mechanics by discussing what GAs process and how they process it.

Suppose we are seeking to find a *solution* to some *problem*. To apply a genetic algorithm to that problem, the first thing we must do is *encode* the problem as an artificial *chromosome* or chromosomes. These artificial chromosomes can be strings of 1s and 0s, parameter lists, or even complex chromosomes, but the key thing to keep in mind is that the genetic machinery will manipulate a finite representation of the solutions, not the solutions themselves.

Another thing we must do in solving a problem is have some means or procedure for evaluating good solutions from bad solutions. This can be as simple as having a human intuitively choose better solutions over worse solutions, or it can be an elaborate computer simulation or model that helps determine what good is, but the idea is that *something* must determine a solution's relative *fitness to purpose* and whatever that is will be used by the genetic algorithm to guide the evolution of future generations.

Having encoded the problem in a chromosomal manner and having devised a means of determining good solutions from bad ones, we prepare to *evolve* solutions to our problem by creating an initial *population* of encoded solutions. The population can be created randomly or by using prior knowledge of possibly good solutions, but either way a key idea is that the GA will search from a population, not a single point.

With a population in place, *selection* and *genetic* operators can work on the population to create a sequence of populations that hopefully will contain more and more good solutions to our problem. 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—this is the survival-of-the-fittest mechanism we impose on our solutions. This can be accomplished in a variety of ways. Weighted roulette wheels can be spun, local tournaments can be held, various ranking schemes can be invoked, but however we do it the main idea is to *prefer better solutions to worse ones.* Of course, if we were to only choose better solutions repeatedly from the original database of initial solutions, we would expect to do little more than fill the population with the best of the first generation. Thus, simply selecting the best is not enough, and some means of creating new, possibly better individuals must be found; this is where the genetic mechanisms come into play.

Recombination is a genetic operator that *combines bits and pieces of parental solutions* to form, new, possibly better offspring. Again, there are many ways of accomplishing this and achieving competent performance does depend on getting the recombination mechanism designed properly, but the primary idea to keep in mind is that the offspring under recombination will not be identical to any particular parent, but will instead *combine parental traits in a novel manner.* By itself, recombination is not all that interesting of an operator, because a population of individuals processed under repeated recombination alone will basically undergo what amounts to a random shuffling of the components as exchanged by the crossover operator.

Where recombination creates a new individual by recombining the traits of two or more parents, mutation acts by simply modifying a a single individual. There are many variations of mutation, but the main idea is that the offspring be identical to the parental individual except that *one or more changes is made to an individual's trait or traits* by the operator. By itself mutation represents a "random walk" in the neighborhood of a particular solution. If done repeatedly over a population

[1] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley

of individuals, we might expect the resulting population to be indistinguishable from one created at random.

3 The Fundamental Intuition

The previous section described the *mechanics* of a genetic algorithm, but it gives us little idea of why these operators might promote a useful search. To the contrary, individually we saw how the operators acting alone were ineffectual, and it is something of an intellectual mystery to explain why such individually uninteresting mechanisms acting in concert might do something useful together. Over the years, 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 together and that of selection and recombination taken together to different *facets of human innovation*, what I will call the *improvement* and *crossfertilizing* types of innovation. We start first with the combination of selection and mutation and continue with the selection-recombination pair.

3.1 Selection + Mutation = Continual Improvement

When taken together, selection and mutation are a form of *hillclimbing* mechanism, where mutation creates variants in the neighborhood of the current solution and selection accepts those changes with high probability, thus climbing toward better and better solutions. Human beings do this quite naturally, and in the literature of total quality management this sort of thing is called *continual improvement* or as the Japanese call it, *kaizen*. When I first introduced the innovation intuition, it was largely based on introspection, but others have had similar thoughts, for example the British author and politician Bulwer-Lytton:

> Invention is nothing more than a fine deviation from, or enlargement on a fine model. ...Imitation, if noble and general, insures the best hope of originality. *E. Bulwer-Lytton*

Although this qualitative description is a far piece from an algorithmic one, we can hear the echo of mutation and selection within these words. Certainly, continuing to experiment in a local neighborhood is a powerful means of improvement, although it will have a tendency to be fairly local in scope, unless a means can be found for intelligently jumping elsewhere when a locally optimal solution is found.

3.2 Selection+Recombination=Innovation

One way of promoting this kind of intelligent jumping is through the combined effect of selection and recombination, and we can start to understand this if we liken their effect to that of the processes of human crossfertilizing innovation. What is it that people do when they are being innovative in a crossfertilizing 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 will be better than either notion taken individually. Again, my first thoughts on the subject were introspective ones, but again others have written along similar veins, for example, the French mathematician Hadamard:

> We shall see a little later that the possibility of imputing discovery to pure chance is already excluded. ...Indeed, it is obvious that the invention or discovery, be it in mathematics or anywhere else, takes place by combining ideas. *J. Hadamard*

Likewise, the French poet-philosopher Valéy had a similar observation:

> 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. *P. Valéy*

Once again, verbal descriptions are far from our more modern computational kind, but something like the innovation intuition has been clearly articulated by others.

With a basic understanding of the mechanics and power of genetic algorithms, we now examine some of the technical lessons of modern GA design and what they can teach us about the processes of innovation.

4 Down and Dirty: Some of the Technical Lessons of GA Design

As I pointed out in the introduction, the primary difficulty of using innovation as an *explanation* or *design metaphor* for GAs is that the processes of innovation are themselves not very well understood. The more interesting possibility is that if the connection between innovation and GAs holds, the design of more effective GAs should construct a mechanistic description of some of the facets of innovation itself. I believe this may be the ultimate lesson of GA theory and design, and in this section I review some of the technical lessons of GA design for our understanding of the mechanics of innovation. Here, I will abandon my concern above for the facet of innovation I called continual improvement and instead will focus on some of the key lessons of designing effective GAs for understanding the selectorecombinative or crossfertilizing type of innovation.

What do GAs process? The primary idea of selectorecombinative GA theory is that genetic algorithms work through a mechanism of *quasi-decomposition* and *recomposition*. John Holland, one of the great early pioneers of evolutionary computation, called well-adapted sets of features that were components of effective solutions *building blocks*, and the basic idea is that GAs (1) implicitly identify building blocks or subassemblies of good solutions and (2) recombine different subassemblies to form very high performance solutions.

BB growth and timing. Another key idea is that BBs or notions exist in a kind of competitive market economy of ideas, and steps must be taken to ensure that the best ones (1) grow and take over a dominant market share of the population and (2) the growth rate can be neither too fast, nor too slow.

BB decision making. Understanding selectorecombinative GAs helps us understand that the decision making among different, competing notions is *statistical* in nature, and that as we increase the population size, we increase the likelihood of making the best possible decisions.

BB identification and exchange. Perhaps the most important lesson of current research in GAs is that the identification and exchange of BBs is the critical path to innovative success.

Hard problems are BB challenging. A final lesson of GA theory for crossfertilizing innovation is that problems that are hard from the standpoint of innovation are problems whose BBs are hard to acquire. This may be because the BBs are deep or complex, hard to find, or because different BBs are difficult to separate, but whatever the difficulty it may be understood in strictly mechanistic terms.

These somewhat technical lessons may be translated into more practical terms, a task taken up in the next section.

5 Lessons for the Real World

Constructing genetic algorithms that work focuses one's mind in a way that qualitative theorizing can never match. As a result, the lessons of GA design for understanding real-world innovation are battle-tested under the severest of conditions. With this in mind, I've tried to abstract some of those lessons in a manner that might make sense in day-to-day business.

There are different modes of innovation. GA research helps us identify some of the different facets of innovation quantiatively, but recognizing that improvement, cross-fertilizing, and possibly other modes of innovation exist can be crucial to designing systems and incentives that encourage them.

The wisdom is in the population. GA research teaches us to respect the generation of an outstanding individual, but ironically the creation of the fine individual solution comes about from the "wise" action of a changing population. The population is not only the original source of good notions, but it is also the testing ground for being sure that the best notions are indeed the best. Moreover, the population is where we "break some eggs to make an omelette," the place where we fail so we may ultimately succeed.

Innovation has a sweetspot. Just like a tennis racket has a sweet spot where even off-center shots play true, innovation is performed well when a number of variables are properly aligned. The frequency of cross-fertilizing events, the severity of choosing the best individuals, and the number of different possibilities that one entertains are the key variables, and the key tradeoff is between frequency of cross-fertilization and severity of choice. Simply put, it is important not to eliminate alternatives too quickly with respect to the rate of generating new ideas. If this is done, the diversity necessary to create something better disappears, making the success rate largely dependent on the local process of continual improvement.

Innovation depends on exchanging the right stuff. GA research teaches us that exchange properly or improperly done is the difference between a solution achieved in a timely fashion or not at all. Effective innovating entities (individuals, organizations, societies) must have mechanisms for trying different *decompositions* of the problem as well as different *classes* of solutions. When individuals try a "new" approach to a problem or when businesses "reorganize" what group reports where, they are trying different ways of slicing the problem in the hopes of making more innovative combinations than was previously possible.

Creativity goes beyond and is more powerful than innovation. I have used the term "innovation" as something of a catchall for invention or improvement, but the study of the cross-fertilization and kaizen modes suggests that there are modes of adaptation that lie beyond the merely innovative. I will use the common term "creativity" to label these and suggest that genetic algorithm research immediately suggests two modes of the creative. The first mode of creativity—*remapping the primitives*—comes about when one changes the *underlying coding of the problem* in an advantageous way. The second mode of creativity—*metaphorical transfer*—comes about when one *transfers a solution* from a different, better understood problem domain to the present one. In either event, GA research teaches us that creativity essentially makes hard problems easier by either directly or indirectly making the building blocks necessary to solve the problem more accessible to the search.

6 Conclusions

This brief paper has reviewed what genetic algorithms are, the fundamental intuition behind their operation, some of the concrete technical lessons of GA research, and how these lessons might impact the way practitioners think about innovation and its facilitation.

Although organizations recognize the fundamental importance of innovation to their continued survival, it is often shrouded in mystery and treated as something of a "soft" subject about which little concrete can be said. This paper suggests that research into genetic algorithms is profoundly changing this picture. Where once mystery and magic were largely found, a healthy dosage of detailed mechanistic understanding now stands. The paper has tried to convey some of the important lessons of the work for the real world, but additional effort is needed to interpret and disseminate this information to a business and organizational audience whose livelihood—whose continued business existence—may depend upon learning and applying these lessons quickly and well.

Acknowledgments

This paper was written while I was on sabbatical at the Section on Medical Informatics at Stanford University. I am grateful to Mark Musen, Russ Altman, and John Koza for inviting me to Stanford.

My contribution to this paper 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, and F49620-97-1-0050. 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 or the U. S. Government.