

The Design of Innovating Machines: A Fundamental Discipline for a Postmodern Systems Engineering

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

1 Introduction

Systems engineering has been the Rodney Dangerfield of engineering disciplines: it gets no respect. On the one hand, everyone now claims to “do systems,” but few are willing to specialize in it (thinking that doing so is by definition an oxymoronic enterprise). Moreover, those who do specialize in systems can’t seem to agree on what it is. Fortunately, a growing number of engineering programs in the US and around the world are looking at systems engineering anew with an eye toward building a consensus on what this paper will call *postmodern systems engineering*.¹ Indeed, the MIT Engineering Systems Division Symposium is an important event toward building that consensus, and our debates today may shape the future of systems engineering research, teaching, and practice in important ways for years to come.

Although the consensus has not yet emerged, it is increasingly clear that it will not be your grandmother’s system engineering. Where once system engineering was largely a Cold War tool for planning, executing, and controlling large-scale space, military, and industrial projects necessitated by the space race, various arms races, and industrial needs, the postmodern consensus seems destined to view systems engineering in a context of democratic, post-industrial capitalist economies (Fuyukama, 1992), where human needs and influences are well integrated into the engineering process.

The last phrase of the last sentence in the preceding paragraph is a tough challenge. To better account for human needs and influences, systems engineering must retool itself and build new models and disciplines. Discussing the full range of disciplines and models needed is beyond the scope of this paper; however, here I suggest that postmodern systems engineering requires a better understanding of the theory and processes of *human innovation*. Systems engineers of times past failed to anticipate the innovative capability of their users, adversaries, and colleagues. By better understanding processes of human innovation, postmodern systems engineers will be better able to (a) harness the innovative capability of their own organizations and (b) anticipate or, at least, better understand the unintended innovative ability of those who will use or try to defeat the system designed.

Creating an engineering understanding of human innovation is a daunting task, and if we were to start *de novo* the odds would be quite long; however, more than twenty years ago, I likened the operation of *genetic algorithms* (GAs) to certain processes of human innovation (Goldberg, 1983). Genetic algorithms are search procedures based on the mechanics of natural genetics and selection (Goldberg, 1989). At the time, my aim was to give a plausible explanation of GA power to new readers in an effort to connect with those who might otherwise find the operation of GAs somewhat suspect. Since that time, I’ve come to realize that the connection between GA mechanics and innovation is much closer than I originally surmised.

And in that story lies the threefold purpose of this paper and the book upon which it is based (Goldberg, 2002):

¹The use of “postmodern” is intended to resonate with the philosophical usage of the term, in which notions of truth depend on the influence of populations of human observers; technically oriented readers may find Searle’s (1995) account with its careful delineation of social and brute facts to be a useful framework.

1. Elucidate the elements of genetic algorithm design theory and GA design that have enabled the creation of *competent* GAs—GAs that solve hard problems quickly, reliably, and accurately.
2. Suggest that these competent genetic algorithms and their successors are the beginnings of a constructive mathematical-computational theory of innovation.
3. Assert that this emerging theory of innovation is important to the practice of postmodern systems engineering.

The paper starts with a brief review of the mechanics of a GA and discusses how different combinations of the basic operators can be thought of as different facets of innovation. It then briefly considers certain elements of a design theory of competent GAs, a theory that also serves as foundation for a mathematical theory of innovation. The paper concludes by considering some of the ways in which this GA-innovation connection can become a critical weapon in the postmodern systems engineer's arsenal.

2 The One-Minute Genetic Algorithmist

I have written elsewhere at length (Goldberg, 1989) about GA basics, and those interested in more detail should consult that or other elementary references. In this section, we quickly review the fundamental mechanics by discussing what GAs process and how they process it.

Suppose we are seeking a *solution* to some *problem*. The first thing we must do to apply a genetic algorithm to that problem is *encode* it as an artificial *chromosome* or chromosomes. These artificial chromosomes can be strings of 1s and 0s, parameter lists, permutation codes, or even complex computer codes, and one of the surprises found over the last decade is how well selectionist schemes do when faced with widely different codings.

Another thing we must do in solving a problem is to have some means or procedure for discriminating good solutions from bad solutions. This can involve the usual elaborate computer simulations or mathematical models that help determine what good is (the standard notion of an *objective* function), or it can be as simple as having a human intuitively choose better solutions over worse ones (what we might call a *subjective* function). It can even be an ecology-like process where different digital species *co-evolve* through an intricate mix of competition and cooperation. However it is done, the idea is that *something* must determine a solution's relative *fitness to purpose* (or *context*), 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 discriminating 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 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 more offspring to better individuals; this is the survival-of-the-fittest mechanism we impose on our solutions. It can be accomplished in a variety of ways. Weighted roulette wheels can be spun, local tournaments can be held, or 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 only to 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 recombination and mutation 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 and 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 undergo what amounts to a random shuffling of extant traits.

Where recombination creates a new individual by recombining the traits of two or more parents, mutation acts by simply modifying 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 are 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 of individuals, we might expect the resulting population to be indistinguishable from one created at random.

3 The Fundamental Intuition

The previous section briefly 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 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, the innovation intuition likens the processing of selection and mutation together and that of selection and recombination taken together to *different facets of human innovation*, what are here called the *improvement* and *cross-fertilizing* 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; 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 wrote about the innovation intuition, it was largely based on introspection, but others have had similar thoughts, for example the British author and politician Bulwer-Lytton (Asimov & Shulman, 1988, p. 118):

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.

Although this qualitative description is distant 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 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. Again, my first thoughts on the subject were introspective ones, but others have written along similar veins, for example, the French mathematician Hadamard (1954, p. 29):

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

Likewise, the French poet-philosopher Valéry had a similar observation (Hadamard, 1954, p. 30):

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.

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

4 The Goals, Decomposition, and Theory of Competent GAs

The innovation intuition is useful in understanding the importance and power of genetic algorithms, but such discussion does not create functional code. Here, we are clearer about the goals, design decomposition, and theory required for the creation of effective genetic algorithms.

4.1 Designing Competent GAs

The primary objective is to design what we call *competent* genetic algorithms. A GA is called competent if it can solve

1. hard problems,
2. quickly,
3. accurately, and
4. reliably.

Each of these can be quantified further, but qualitatively, hard problems are those that have large subsolutions that must be discovered whole, badly scaled subsolutions, many different local optima, a high degree of subsolution interaction, or a lot of external noise or stochasticity. In short, we are interested in designing effective solvers for the class of nearly decomposable problems (Simon, 1969). Speed, accuracy, and reliability requires that we get to near-global or high-quality solutions in times that grow as a polynomial function of the number of decision variables with high probability.

4.2 The Elements of GA Design Theory

With the goals of GA design in place, we are in a position to articulate the elements of the design decomposition to be followed. Specifically, we decompose the problem of designing a competent selectorecombinative GA into the following seven subproblems:

1. Know what GAs process—building blocks.
2. Know thy BB challengers—building-block-wise difficult problems.
3. Ensure an adequate supply of raw BBs.
4. Ensure increased market share for superior BBs.
5. Know BB takeover and convergence times.
6. Make decisions well among competing BBs.
7. Mix BBs well.

Several of these items are briefly reviewed. More detail is available in the original reference (Goldberg, 2002).

Understanding BBs. The notion of building blocks has been remarkably controversial considering that it underpins most of the success in human thought over the last 300 years. Descartes (1994) articulated the fundamental need to decompose big problems into little problems in his *Discourse on Method* centuries ago,

and human knowledge and thought hasn't been the same ever since. In GAs, we simply mechanize Cartesian decomposition and raise it to an art form; a key move is to understand what the chunks of decomposition might be. Holland (1975) started us on a rigorous road to identifying these chunks mathematically in the 60s and 70s, and those efforts continue today.

Design adversarial test functions. Once we can talk about decomposition, it makes sense to talk about those problems that *thwart* our ability to decompose. This leads us naturally to such matters as *deception* and a fuller theory of problem difficulty is presented elsewhere (Goldberg, 2002). Although much of the literature of computation looks at test cases in this or that problem, a key move here is to look at *boundedly adversarial* test cases. By knowing the theory of operation of our algorithms, we can design test functions that exploit weakness. In a game-theoretic sense, if we develop algorithms that scale well in the face of such adversaries, we should expect the procedures to perform as well or faster on problems easier than those at the limits of the design envelope.

Market share, timing, supply, decision-making, mixing, and all that. Once we focus on building blocks, and once we understand the adversaries we must face, we treat BBs as a kind of material quantity that must be transported through space and time. Upstream, we ensure that we have enough raw BBs to solve the problem at hand (the supply question). We then make sure that selection is pushy enough and that the innovation operator is safe enough to ensure that BB market share grows with time (the schema theorem or market share question). Thereafter, we consider how long convergence takes on average (convergence time), and ensure that the pool of choices is sufficiently rich to permit good solutions (the decision question). Finally, we make sure that different building blocks come together on the same string through effective exchange (BB mixing).

From design theory to competent GA design. If all of these steps are performed successfully, we have designed a GA that scales well (often subquadratically) and life is good. If one or more of the key requirements are violated, the GA fails, and in worst cases the subquadratic performance turns exponential. Over the years, different mechanisms have been developed to ensure competence. Interestingly, the mechanisms are very different from one another when viewed from an algorithmic point of view. Yet, when viewed through the lens of theory, all competent GAs obey the same *principles* of operation.

5 Why is this Important to Systems Engineering?

Many engineers approach genetic algorithms with the need to solve a specific problem, and to that end, the use of GAs now cuts across many if not most areas of human endeavor. Like the application of human innovation itself, GAs put few prior restrictions on the kinds of problems that can be tackled, and many practitioners have gone from viewing GAs as solvers of last resort to solvers of most frequent resort.

But postmodern systems engineers aspire to account for human needs, human adversaries, and humans in the loop, and accounting for these matters requires an engineering theory of innovation. Of course, loose talk of innovation is all around us. Modern companies strive to be increasingly innovative in a time of global competition. Modern social science understands that one of the primary failings in setting social policies is the inability to anticipate the innovation of human actors. This inability is so strong it is raised to the level of a "law," the *law of unintended consequences*. Against this backdrop, much of the discussion of innovation is qualitative in nature, and our understanding of innovation in humans and their organizations is unsatisfying, and not up to the usual standards of engineering knowledge.

Yet, this paper asserts that the situation is already more interesting than this. If the connection between genetic algorithms and innovation holds up, and if our understanding of GAs is markedly advanced because of our understanding of competence theory and its implications for scalable GAs (which I assert, it is), then the theory of competent GAs and their invention represents a mathematical-computational theory of key facets of human innovation.

This is an important prospect for postmodern systems engineering. Competent GAs and competence theory can be used as a theory of key facets of human innovation. That theory can then be used to

1. create high-performance problem solvers;

2. create intelligent, adaptive agents as part of system simulations;
3. create qualitative and quantitative theories of human innovation that can be tested empirically;
4. help design large-scale systems that encourage virtuous innovation and resist the innovative efforts of adversaries;
5. help understand the limits of system design with respect to large populations of innovators.

The first item is the usual role of genetic algorithms in systems engineering and is not discussed further. Others at this workshop have pointed out the importance of agent-based simulation in postmodern systems engineering, and the theories discussed herein can help provide such agents with effective, efficient adaptation capability. Using these ideas to inform innovation theory, to design large-scale systems more generally, or to understand the limits of systems design in the face of large numbers of innovating actors is just beginning, but the need for such knowledge is great and the prospects for important advances are now good.

6 Conclusions

The paper started by making a distinction between Cold War systems engineering and what it called postmodern systems engineering. Postmodern systems engineering is increasingly concerned with the role of human needs, interaction, and innovation in systems design, and this emphasis directs postmodern systems engineering in new directions.

The paper then briefly considered the connection between GAs that scale well—so-called competent GAs—and human innovation, and it has suggested that the theory and practice of competent GAs forms a kind of mathematical-computational theory of important facets of human innovation. Given the aspirations of postmodern systems engineering in understanding human-system interaction, understanding innovation in mathematical-computational terms should give an important boost to systems design, both qualitatively and quantitatively.

Although such understanding will lead to a positive approach to design for innovation, it will also help us understand the limits of centralized control when faced with a population of innovating actors. The utility of the theory and design of competent GAs for continued technological advance and scientific understanding is not yet known, but first returns are promising. The parting hope of this paper is that many more will join the quest for using and understanding these ideas and their offspring.

Acknowledgments

This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-00-0163 and F49620-03-1-0129, and by the Technology Research, Education, and Commercialization Center (TRECC), a program of the University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research under grant N00014-01-1-0175. 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 those of the author 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 Office of Naval Research, or the US Government.

References

- Asimov, I., & Shulman, J. A. (Eds.) (1988). *Isaac Asimov's book of science and nature quotations*. New York, NY: Weidenfeld & Nicolson.
- Descartes, R. (1994). A discourse on the method of rightly conducting the reason, and seeking truth in the sciences [Veitch, J. (trans.)]. In Sorell, T. (Ed.), *A Discourse on Method: Meditations and Principles* (pp. 3–57). London, UK: Everyman. (Original work published 1637).

- Fuyukama, F. (1992). *The end of history and the last man*. New York: NY: Free Press.
- Goldberg, D. E. (1983). *Computer-aided gas pipeline operation using genetic algorithms and rule learning*. Doctoral dissertation, University of Michigan, Ann Arbor, MI. (University Microfilms No. 8402282).
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.
- Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms*. Boston, MA: Kluwer Academic.
- Hadamard, J. (1954). *The psychology of invention in the mathematical field*. New York, NY: Dover.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
- Searle, J. R. (1995). *The construction of social reality*. New York, NY: Free Press.
- Simon, H. A. (1969). *Sciences of the artificial*. Cambridge, MA: MIT Press.