Designing Biomorphs with an Interactive Genetic Algorithm

Joshua R. Smith*
jrs@cs.williams.edu
Department of Computer Science
Williams College
Williamstown, MA 01267

Abstract

We describe an interactive version of the Genetic Algorithm which can be used to evolve Fourier series based "biomorphs." The user plays the role of a binary-valued objective function. The usual Genetic operators of Selection, Mutation, and Crossover are applied normally. We discuss the Interactive Genetic Algorithm's use as a teaching tool and in domains in which measures of utility are difficult or impossible to specify mathematically. We argue that the evolution of biomorphs can be considered an ergonomic design problem, and with the evolution of biomorphs as an example, we argue that a large class of design problems can be cast as "imprecise optimization" and "solved" with Interactive Genetic Algorithms. Thus IGAs can be used for applications that fall somewhere between AI, optimization, and CAD.

1 Introduction

When does a curve look like a bug? "Looks-like-a-bug" is not the sort of relation that can easily be described in mathematical terms. Thus the Genetic Algorithm, which usually requires a precisely formulated fitness function, may not seem particularly "fit" for evolving bug-shaped curves. But if the objective function is replaced by a human expert, the GA can be applied in this and other imprecise domains. For example, if one were trying to evolve a model of a material with specific "squishy" or "fleshy" properties, it might be simpler to have a human expert tell the system which samples "felt" right than to try to specify mathematically the desired "feel" of the material.

A curve "looks-like-a-bug" when it interfaces with the human visual system in some particular but difficult to specify way, not when it satisfies some easily enumerated set of mathematical criteria. Thus the problem of evolving curves which look like bugs can be considered a "human-factors" or *ergonomic* design problem. Simon[7] pointed out that certain design problems, particularly in engineering, can be cast as optimization problems. Interactive Genetic Algorithms enlarge the class of problems that can be treated in this way. Ergonomic design is particularly well suited to IGA techniques because at some level, humans are the only possible measure of the quality of human-factors design. We maintain that the problem of evolving a curve which looks like a bug has much in common with other ergonomic design problems and that it may often be feasible to treat relations such as "feels-comfortable" in the same way we treat "looks-like-a-bug" here.

Like many versions of the Genetic Algorithm, the Interactive Genetic Algorithm has evolution-inspired "select-and-mutate" ancestors which do not achieve the GA's celebrated implicit parallelism[5] [4] because they do not maintain a large breeding population and do not use the crossover operator. The most famous of these ancestors is of course Richard Dawkins' *Blind Watchmaker,* which used a select-and-mutate approach to evolve tree-based forms which look surprisingly life-like. Dawkins christened these life-like shapes *biomorphs.* Obviously, we have inherited more than just the *term* biomorph from Dawkins: he originated the project of using interactive evolution techniques to grow them. Our variation on the process is the crossover operator and the maintenance of a large breeding population: in Dawkins' program, the next generation inherits its genetic material from a single parent in the current generation[2][1].

In a more practical vein, Oppenheimer [6] used interactive select-and-mutate methods to evolve impressive three dimensional images of trees (the biological variety, not computer science "trees"). Like Dawkins, Oppenheimer used a breeding population of size one and did not employ the crossover operator.

*Emmanuel College, Cambridge England, as of Oct. 1991

Yet Holland's Schema Theorem holds as well for interactive GAs in which the objective function has been replaced by a human operator as it does for standard GAs. The difference is simply that it is impossible to specify *a prioi* which schemata will have above average fitness. Those short, low-order schemata with above-average fitness (no longer a precisely defined quantity) will still receive an exponentially increasing number of trials as long as the human operator behaves consistently. In interactive evolution problems, implicit parallelism is especially desirable, since time spent evolving a solution is expensive human time. Thus it makes sense to use the Genetic Algorithm for interactive evolution problems.

2 BUGS

Our example system for solving an interactive evolution problem enables the user to design biomorphs with the Genetic Algorithm. A biomorph's genotype consists of two sets of real numbers[1] which serve as Fourier coefficients in parametric equations specifying X and Y coordinates as a function of the parameter t. We call these two sets of numbers chromosomes. In the equation for $X(t)$, we let the coefficients of the sin terms equal zero; in the $Y(t)$ equation, we let the cos coefficients equal zero. That is why our curves are specified with two sets of numbers, not the four that would generally be required to specify two Fourier series. To put it succinctly, the first chromosome is the set of A_is and the second is the set of B_is in parametric equations of the form

$$X = \sum_{i=0}^{n} A_i \cos it$$

and

$$Y = \sum_{i=0}^{n} B_i \sin it.$$

We "grow" biomorphs from their genetic material, the Fourier coefficients, by graphing these parametric equations. Figure 1 shows some sample biomorphs. Their bilateral symmetry is due to the periodicity properties of the sin and cos functions.

We also graph each biomorph's genetic material below its 'portrait'. For each chromosome there is a baseline, on which genes with value zero fall. Genes above the line are positive; those below are negative. The graphs of the genes are visible below each biomorph in the Figure.

[1] Although we used real codings for the genes, there was no particular reason for this, and we could have just as easily used binary codings. Considering our earlier arguments about implicit parallelism, we probably should have.

3 An IGA in Action

As in most Genetic Algorithms, the population is initialized randomly. Then the user indicates which biomorphs may reproduce by clicking on-screen buttons. When the user gives the signal, the next generation is produced by fairly standard GA operators. The biomorphs which the user has marked as "fit to breed" all receive equal, positive fitness values; those not marked fit by the user receive fitness zero. At this point we chose pairs for breeding using the standard "roulette wheel" selection operator, stochastic sampling with replacement. Since the user is playing the role of a binary objective function, not of the selection operator, s/he can specify that certain biomorphs do *not* breed but cannot ensure that particular biomorphs *will* breed, and cannot specify which will breed with which. If the user does not indicate that any biomorphs are fit enough to breed, we assume they are all equally good; it is as if the user had indicated that all the biomorphs were fit.

We use one-point crossover, though we perform it twice, once per chromosome. Since we use real codings, we perform mutation by probabilistically adding Gaussian noise to each gene. The variance of the noise is specified as a parameter to the program.[2]

4 The IGA as a Teaching Tool

Using an interactive GA is a good way to gain an intuitive understanding of some features of GA operation. Since the genome is graphed on screen, the user can get some feeling for the operation of the crossover and mutation operators. The user may also acquire an understanding of stochastic selection and its perils: organisms marked fit by the user may nonetheless be passed up by chance. These stochastic sampling errors lead to the phenomena of genetic drift and premature convergence. If the user chooses a very small breeding population, the population converges almost immediately, which is not surprising to most people. But even if all biomorphs are given a chance to breed every time, that is, even in the absence of selective pressure, stochastic sampling errors will cause the population to converge after a time, a phenomenon known as genetic drift.[3] This may also teach a valuable lesson about biological evolution: not all inherited traits are adaptive. IGAs are useful both for improving one's own understanding of these phenomena and for explaining them to others.

[2] We would like to reiterate that these deviations from GA orthodoxy had nothing to do with the interactivity of our system; we could have just as easily used the standard crossover and mutate operators, or any others for that matter.

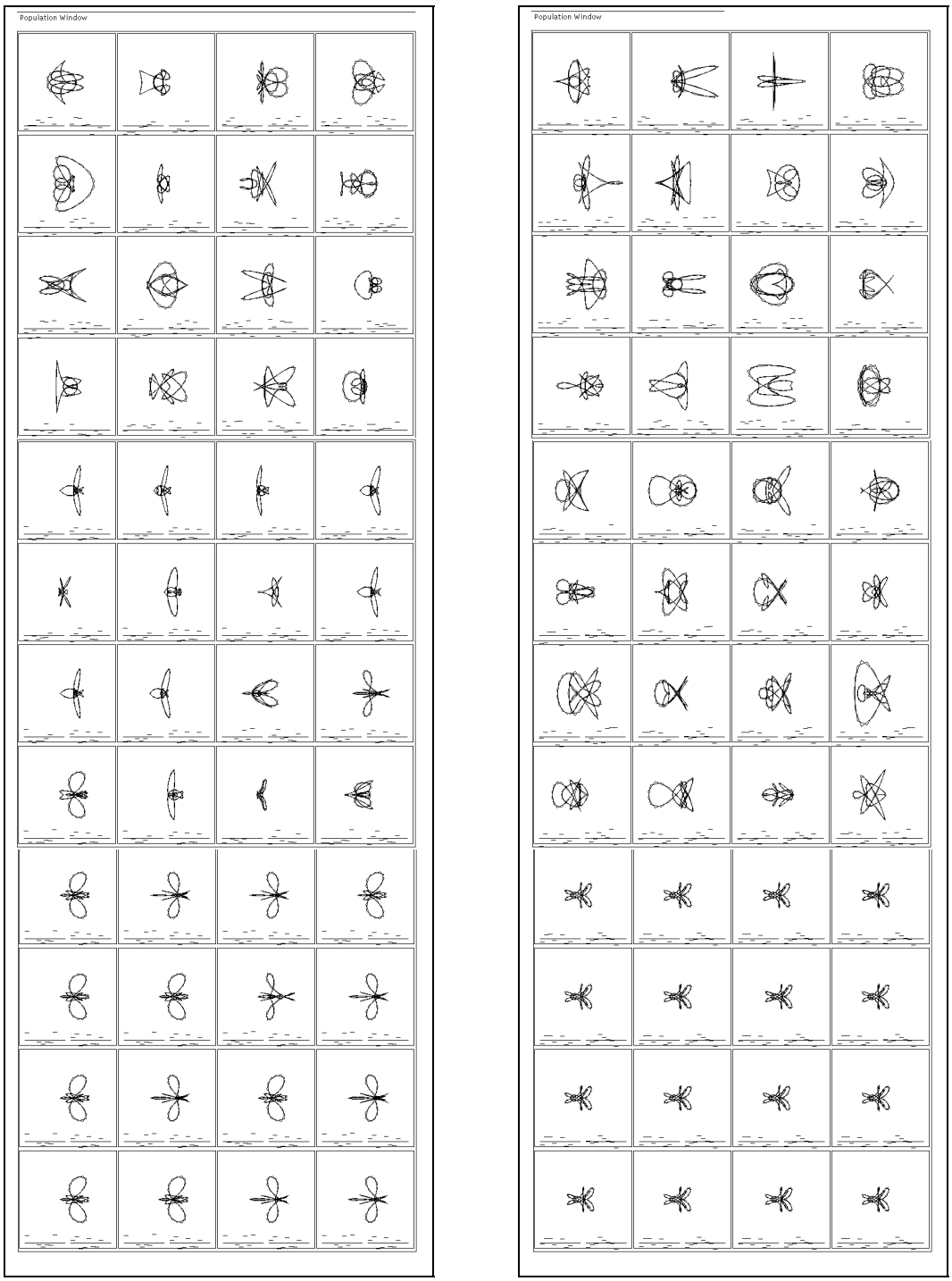

Figure 1: INTERACTIVELY GUIDED EVOLUTION of two biomorph populations. Below each biomorph is a graph of its genes. Each small horizontal segment represents a single gene. The large horizontal segments indicate the origin: a gene above its origin line has positive value; genes below are negative. In both trials shown here, the initial population (*top*) is generated randomly. After several generations of evolution with an interactive genetic algorithm, the curves become more bug-like. Diversity diminishes throughout this process, but the population does not converge immediately (*middle*). The time to convergence depends strongly on how selective the user is, that is, on the fraction of each population designated fit to breed. The final form to which the population converges (*bottom*) depends on the user's preference, on stochastic factors such as the make up of the initial population, and on the Fourier series-based "embryology" function which maps genes into curves.

5 The IGA and Imprecise Optimization

But, as we have already suggested, the IGA's value is not only pedagogic. Using an "interactive" or "human-expert" fitness function may allow the GA to be applied in otherwise inaccessible domains. We have already given a few examples of problems for which we think an IGA approach may be appropriate: the evolution (or "design") of biomorphs and the evolution of models of "squishy" materials, materials whose properties are difficult to specify quantitatively. We will call problems such as these "imprecise optimization" problems.

We will now try to map more precisely the class of IGA-solvable problems. Most of the conditions for GA applicability, with the notable exception of the need for a precisely formulated objective function, are also conditions for IGA applicability. It must be possible to formulate the problem as a search through a parameter space. That means the relevant parameters must be known. This is an important constraint: the need for this knowledge is what separates true design from mere optimization. A designer must discover relevant parameters; an optimizer, given this knowledge, must find good values for them. Again, IGAs, since they are based on the GA, search given parameter spaces. IGAs are optimizers in this sense.

Interactivity imposes an additional constraint on the class of IGA solvable problems: it must be possible to produce a new generation in near real time. While it might be acceptable to let a standard GA chug away for weeks at a time on a problem, a human is liable to lose patience if called upon periodically to babysit an IGA process over a long period of time. The IGA will be most useful when the expert or designer can see the next generation almost immediately (hence the name *interactive* GA).

So the IGA is likely to be applicable to optimization problems in which i) candidate solutions can be generated in near real time and ii) the utility of candidate solutions can be compared by humans but not (practically) by means of a precisely specified formula. This encompasses a class of design problems broader than engineering design (in which candidate designs can be compared on the basis of numbers like weight and cost) but which does not include all design problems, since the parameter space must be known in advance. Again, this second condition suggests that the IGA may be particularly well suited for ergonomic design.[3]

In these contexts, the IGA starts to sound more like an advanced CAD tool than an optimization technique. And perhaps that is where it will prove most useful: as a semi-intelligent assistant to suggest ideas the designer may not have thought of, and to help the designer focus the design by throwing away bad ideas and presenting good ones for frequent iterative refinement.

Acknowledgements

This work supported by the Bronfman Science Foundation. Thanks to Donald House and Duane Bailey of Williams College for guidance, and to Michael Donofrio and William Lenhart for their comments on this paper.

Note:
The software discussed here is available via anonymous ftp at the Santa Fe Institute (`ftp ftp.santafe.edu /pub/OLD/Users/jrs/BUGS/BUGS.tar.Z`). It runs on Sun workstations under Suntools or under the X Window System with XView.

References

[1] Richard Dawkins. *The Blind Watchmaker.* Longman, Harlow, 1986.

[2] Richard Dawkins. The evolution of evolvability. In Christopher G. Langton, editor, *Artificial Life: The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, Redwood City, California, 1989. Addison-Wesley.

[3] David E. Goldberg. Finite markov chain analysis of genetic algorithms. In John J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[4] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1989.

[5] John H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, 1975.

[6] Peter Oppenheimer. The artificial menagerie. In Christopher G. Langton, editor, *Artificial Life: The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, Redwood City, California, 1989. Addison-Wesley.

[7] Herbert A. Simon. *The Sciences of the Artificial.* MIT Press, Cambridge, Massachusetts, second edition, 1985.

[3] For an example of ergonomic design that is close to the hearts of computer scientists, consider user interface design. IGAs, perhaps in conjunction with classifier systems, might find a home in adaptive user interfaces in which designing the interface became an ongoing process of interaction between the user and the machine. In the competition to do the user's bidding, some agents would make mistakes. The user would indicate that an error had occurred, and the responsible agents would receive low fitness ratings. In this way, one might evolve agents which can, to some extent, do "what-I-mean".