Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2008), *22*, 183–184. Printed in the USA.

doi:10.1017/S0890060408000115

GUEST EDITORIAL

Introduction to the Special Issue on genetic programming for human-competitive designs

LEE SPECTOR
School of Cognitive Science, Hampshire College, Amherst, Massachusetts, USA

Readers of this journal are familiar with many ways in which computers aid designers, often serving as design tools or even as design assistants. This Special Issue of *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* concerns recent work that aims to further expand the function of computers in design, to the point at which computers are useful as designers in their own right.

The specific technique on which we focus is genetic programming, a specialization of genetic algorithms (Holland, 1992) to the problem of automatically synthesizing executable computer programs (Koza, 1992). In genetic programming, as in genetic algorithms more generally, problems are solved using processes borrowed from biological evolution including variation, recombination, and selection. In genetic programming, more specifically, the entities that are varied, recombined, and selected (and, hence, evolved) are executable computer programs.

Genetic algorithms work by initially generating large populations of random individuals (candidate designs, sets of parameters, or instances of whatever other type of structure we are seeking) and by assessing the quality of the individuals in an automated way. Although few of the initial individuals will ever be any good, some of them will probably be better than others. The algorithm proceeds by selecting some of the better individuals to serve as "parents" of the next generation, and by producing "children" using processes of mutation and recombination that are modeled loosely on biological genetics. Children are often worse than their parents, but sometimes they are better, and over many generations it is often possible to produce descendants of very high quality. In many cases, including those documented in this Special Issue, this fully automated process eventually produces solutions that rival or surpass those produced by human experts.

Genetic algorithms have been used previously in computer-aided design, and they have been the topic of special issues of other design-oriented journals (Renner, 2003). The more specific genetic programming technique, in which executable programs are generated and evolved, provides new opportunities. One advantage of genetic programming is that it can often simplify the process of evolving *entire* designs, including not only the design's numerical parameters but also its overall structures or topologies. This is often accomplished by evolving a program which, when executed, builds (or "develops" from an "embryo") a complete entity of the desired sort. The viability of this developmental approach (see Hornby et al., 2007) is demonstrated by the production of numerous results, many of which are documented in this Special Issue, that are "human-competitive" as measured by objective criteria such as patents and publications (Koza et al., 2000).

In the first article of this issue John R. Koza provides a tutorial on the genetic programming techniques that are used, with some variations, in the remainder of the articles in this issue and in the field more broadly. He also describes criteria for "human-competitiveness" along with 28 published human-competitive results of genetic programming.

The next three articles show how genetic programming can be used to automatically synthesize human-competitive designs in the field of mechanical engineering. Hod Lipson applies the technique to a classical kinematic challenge of the 19th century, the "straight line problem." Jianjun Hu, Erik D. Goodman, Shaobo Li, and Ronald Rosenberg turn to a more modern challenge, the design of passive vibration absorbers that perform better than the standard passive vibration absorbers invented in 1911. In doing so they also demonstrate how genetic programming can work with bond graphs, an energy-based representation scheme that is used by many engineers to model physical systems. The bond graph approach is further expanded in the next article by Jiachuan Wang, Zhun Fan, Janis P. Terpenny, and Erik D. Goodman to demonstrate the evolution of systems with both passive and active components.

In the next two articles we see how genetic programming can be applied to problems in two different areas of electromagnetic design, antennas and optics. Jason D. Lohn,

Reprint requests to: Lee Spector, School of Cognitive Science, Hampshire College, Amherst, MA 01002-3359. E-mail: lspector@hampshire.edu

Gregory S. Hornby, and Derek S. Linden present their work on the evolution of a human-competitive design for an antenna that was deployed on a NASA spacecraft in 2006. Their system produced the first evolved hardware in space and the first evolved antennas to be deployed anywhere; in their article they describe how this was accomplished, and compare their results to those of a human design team. Then John R. Koza, Sameer H. Al-Sakran, and Lee W. Jones describe how they used genetic programming to automatically synthesize complete designs for four optical lens systems that duplicate the functionality of previously patented lens systems. They detail the ways in which the resulting designs infringe on existing patents, or, more frequently, represent novel solutions to long-standing design problems. They argue that their results are human-competitive, that they demonstrate the "routineness" of human-competitive design by genetic programming, and that the rate at which such designs are produced should increase as computers become more powerful and less expensive.

The final two articles describe applications in the small but growing field of quantum computing. These applications are of special interest because the bizarre and counterintuitive nature of quantum mechanics makes human design particularly challenging, thereby increasing the potential utility of automated design systems. My own contribution with coauthor Jon Klein outlines ways in which developmental genetic programming can be applied to quantum computing applications and also demonstrates the invention of a new, better than classical quantum circuit for the two-oracle "AND/OR" problem. Ralph Stadelhofer, Wolfgang Banzhaf, and Dieter Suter then describe several innovations in this area, including improvements to the efficiency of fitness tests, enhancements that enable the evolution of programs for "ensemble" quantum computers, and examples of applications to new kinds of quantum computing problems.

Mechanical, electromagnetic, and quantum system design are only a few examples of areas in which genetic programming can be to applied to design problems. Applications to many other areas, ranging from music to robotics, are documented in the growing literature of the field (see http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html). The generality of the method, which derives from the fact that many design problems can be expressed as searches for a computer programs, bodes well for the application of genetic programming to other areas of design in the future.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants 0308540 and 0216344. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA: MIT Press. (First edition 1975)

Hornby, G.S., Kumar, S., & Jacob, C. (2007). Editorial introduction to the special issue on developmental systems. *Genetic Programming and Evolvable Machines 8(2)*, 111–113.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Koza, J.R., Keane, M.A., Yu, J., Bennett III, F.H., & Mydlowec, W. (2000). Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines 1(1/2)*, 121–164.

Renner, G. (2003). Genetic algorithms in CAD. *Computer-Aided Design 35(8)*, 707–708.

Lee Spector is a Professor of computer science at Hampshire College in Amherst, MA. He received a BA in philosophy from Oberlin College and a PhD from the Department of Computer Science at the University of Maryland. His areas of interest include evolutionary computation, quantum computation, and the intersections between computer science, cognitive science, evolutionary biology, and the arts. He is a member of the ACM Special Interest Group on Evolutionary Computation (SIGEVO) Executive Committee and is a Fellow of the International Society for Genetic and Evolutionary Computation. He serves on the editorial boards of *Genetic Programming and Evolvable Machines* and *Evolutionary Computation.*