Routine Human-Competitive Machine Intelligence by Means of Genetic Programming

John R. Koza
Stanford University
Stanford, California
koza@stanford.edu

ABSTRACT

Genetic programming is a systematic method for getting computers to automatically solve a problem. Genetic programming starts from a high-level statement of what needs to be done and automatically creates a computer program to solve the problem. This summary paper makes the points that (1) genetic programming now routinely delivers high-return human-competitive machine intelligence; (2) genetic programming is an automated invention machine; and (3) genetic programming has delivered a progression of qualitatively more substantial results in synchrony with five approximately order-of-magnitude increases in the expenditure of computer time.

1 Background on Genetic Programming

One of the central challenges of computer science is to get a computer to solve a problem without explicitly programming it to do so. Paraphrasing Arthur Samuel—founder of the field of machine learning—this challenge (1959) is

How can computers be made to do what needs to be done, without being told exactly how to do it?

Genetic programming starts from a high-level statement of what needs to be done and automatically creates a computer program to solve the problem.

Genetic programming uses the Darwinian principle of natural selection along with analogs of recombination (crossover), mutation, gene duplication, gene deletion, and mechanisms of developmental biology to breed an ever-improving population of programs. Genetic programming has been successfully applied to a wide variety of problems from numerous different fields (Koza 1992; Koza 1994; Koza, Bennett, Andre, and Keane 1999; Koza, Keane, Streeter, Mydlowec, Yu, and Lanza 2003).

For additional information, visit www.genetic-programming.com.

2 Genetic Programming Now Routinely Delivers High-Return Human-Competitive Machine Intelligence

We say that a result is "human-competitive" if it satisfies one or more of the eight criteria in table 1. These eight criteria have the desirable attribute of being at arms-length from the fields of artificial intelligence, machine learning, and genetic programming. That is, a result cannot acquire the rating of "human-competitive" merely because it is considered interesting by researchers *inside* the specialized fields that are attempting to create machine intelligence. Instead, a result produced by an automated method must earn the rating of "human-competitive" *independent* of the fact that it was generated by an automated method.

Based on this definition, there are now 36 instances where genetic programming has produced a human-competitive result (table 2).

Table 1 Eight criteria for human-competitiveness

	Criterion
A	The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.
B	The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.
C	The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.
D	The result is publishable in its own right as a new scientific result—independent of the fact that the result was mechanically created.
E	The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.
F	The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.
G	The result solves a problem of indisputable difficulty in its field.
H	The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

Table 2 Thirty-six human-competitive results produced by genetic programming

	Claimed instance	Basis
1	Creation of a better-than-classical quantum algorithm for the Deutsch-Jozsa "early promise" problem	B, F
2	Creation of a better-than-classical quantum algorithm for Grover's database search problem	B, F
3	Creation of a quantum algorithm for depth-two AND/OR query problem that is better than previously published results	D
4	Creation of a quantum algorithm for the depth-one OR query problem that is better than any previously published result	D
5	Creation of a protocol for communicating information through a quantum gate previously thought not to permit it	D
6	Creation of a novel variant of quantum dense coding	D
7	Creation of a soccer-playing program that won its first two games in the Robo Cup 1997 competition	H
8	Creation of a soccer-player ranked in middle of field of 34 human-written programs in Robo Cup 1998 competition	H
9	Creation of four different algorithms for the transmembrane segment identification problem for proteins	B, E
10	Creation of a sorting network for seven items using only 16 steps	A, D
11	Rediscovery of the Campbell ladder topology for lowpass and highpass filters	A, F
12	Rediscovery of the Zobel "*M*-derived half section" and "constant *K*" filter sections	A, F
13	Rediscovery of the Cauer (elliptic) topology for filters	A, F
14	Automatic decomposition of the problem of synthesizing a crossover filter	A, F
15	Rediscovery of a recognizable voltage gain stage and a Darlington emitter-follower section of amplifier and other circuits	A, F
16	Synthesis of 60 and 96 decibel amplifiers	A, F
17	Synthesis of analog computational circuits for squaring, cubing, square root, cube root, and Gaussian functions	A, D, G
18	Synthesis of a real-time analog circuit for time-optimal control of a robot	G
19	Synthesis of an electronic thermometer	A, G
20	Synthesis of a voltage reference circuit	A, G
21	Creation of a cellular automata rule for the majority classification problem that is better than the Gacs-Kurdyumov-Levin (GKL) rule and all other known rules written by humans	D, E
22	Creation of motifs that detect the D–E–A–D box family of proteins and the manganese superoxide dismutase family	C
23	Synthesis of topology for a PID-D2 (proportional, integrative, derivative, and second derivative) controller	A, F
24	Synthesis of an analog circuit equivalent to Philbrick circuit	A, F
25	Synthesis of a `NAND` circuit	A, F
26	Simultaneous synthesis of topology, sizing, placement, and routing of analog electrical circuits	A. F, G
27	Synthesis of topology for a PID (proportional, integrative, and derivative) controller	A, F
28	Rediscovery of negative feedback	A, E, F, G
29	Synthesis of a low-voltage balun circuit	A
30	Synthesis of a mixed analog-digital variable capacitor circuit	A
31	Synthesis of a high-current load circuit	A
32	Synthesis of a voltage-current conversion circuit	A
33	Synthesis of a Cubic function generator	A
34	Synthesis of a tunable integrated active filter	A
35	Creation of PID tuning rules that outperform the Ziegler-Nichols and Åström-Hägglund tuning rules	A, B, D, E, F, G
36	Creation of non-PID controllers that outperform the Ziegler-Nichols or Åström-Hägglund tuning rules	A, B, D, E, F, G

We define the *AI ratio* (the "artificial-to-intelligence" ratio) of a problem-solving method as the ratio of that which is delivered by the automated operation of the *artificial* method to the amount of *intelligence* that is supplied by the human applying the method to a particular problem. Each of the 36 results in table 2 is a human-competitive result and therefore represents a high amount of "A." Only a *de minimus* amount of "I" is contained in the human-supplied primitive ingredients available to the to-be-evolved computer program, the human-supplied fitness measure encapsulating the high-level statement of what needs to be done, and the human-supplied control parameters and termination procedures. In view of the high amount of "A" in the numerator and the small amount of "I" in the denominator, we can see that genetic programming yielded a high return (i.e., a high AI ratio) for these results.

"Routineness" means a result is general and that relatively little human effort is required to get the method to successfully handle new problems within a particular domain and to successfully handle new problems from a different domain.

Using these definitions, the results in table 2 establish that genetic programming now routinely delivers high-return human-competitive machine intelligence.

3 Genetic Programming Is an Automated Invention Machine

There are now 23 instances where genetic programming has duplicated the functionality of a previously patented invention, infringed a previously issued patent, or created a patentable new invention. Specifically, there are 15 instances where genetic programming has created an entity that either infringes or duplicates the functionality of a previously patented 20th-century invention, six instances where genetic programming has done the same with respect to a previously patented 21st-century invention, and two instances where genetic programming has created a patentable new invention.

4 Progression of Qualitatively More Substantial Results Produced in Synchrony with Increasing Computer Power

Table 3 lists the five computer systems used to produce our group's reported work on genetic programming in the 15-year period between 1987 and 2002. Column 7 shows the number of human-competitive results (table 2) generated by each system. Table 3 shows the following:

- There is an order-of-magnitude speed-up (column 4) between each successive computer system in the table. Note that, according to Moore's law, exponential increases in computer power correspond approximately to constant periods of time.
- There is a 13,900-to-1 speed-up (column 5) between the fastest and most recent machine (the 1,000-node parallel computer system) and the slowest and earliest computer system in table 3 (the serial LISP machine).
- The slower early machines generated few or no human-competitive results, whereas the faster more recent machines have generated numerous human-competitive results.

Four successive order-of-magnitude increases in computer power are explicitly shown in table 3. An additional order-of-magnitude increase was achieved by making extraordinarily long runs on the largest machine in table 3 (the 1,000-node Pentium® II parallel machine). The length of the run that produced one of our patentable new inventions (i.e., a genetically evolved general-purpose controller) was 28.8 days—almost an order-of-magnitude increase (9.3 times) over the 3.4-day average for our group's other runs. If this final 9.3-to-1 increase is counted as an additional speed-up, the overall speed-up is 130,660-to-1.

Table 3 Human-competitive results produced by genetic programming with five computer systems

System	Period	Petacycles (10^{15}) per day	Speed-up	Speed-up	Used for work in book	Human-competitive results
Serial Texas Instruments LISP machine	1987–1994	0.00216	1 (base)	1 (base)	*Genetic Programming I* and *Genetic Programming II*	0
64-node Transtech transputer parallel machine	1994–1997	0.02	9	9	A few problems in *Genetic Programming III*	2
64-node Parsytec parallel machine	1995–2000	0.44	22	204	Most problems in *Genetic Programming III*	12
70-node Alpha parallel machine	1999–2001	3.2	7.3	1,481	A minority (8) of problems in *Genetic Programming IV*	2
1,000-node Pentium II parallel machine	2000–2002	30.0	9.4	13,900	A majority (28) of the problems in *Genetic Programming IV*	12

These five order-of-magnitude increases in computing power correspond closely with the following progression of qualitatively more substantial results produced by genetic programming:

- toy problems,
- human-competitive results not related to patented inventions,
- 20^{th}-century patented inventions,
- 21^{st}-century patented inventions, and
- patentable new inventions.

This progression demonstrates that genetic programming is able to take advantage of the exponentially increasing computational power made available by iterations of Moore's law. This progression of results suggests that genetic programming may deliver increasingly more significant results in the future.

References

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

Koza, John R. 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: MIT Press.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1999. *Genetic Programming III: Darwinian Invention and Problem Solving*. San Francisco, CA: Morgan Kaufmann.

Koza, John R., Keane, Martin A., Streeter, Matthew J., Mydlowec, William, Yu, Jessen, and Lanza, Guido. 2003. *Genetic Programming IV. Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers.

Samuel, Arthur L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development.* 3(3): 210–229.