Genet Program Evolvable Mach (2010) 11:251–284
DOI 10.1007/s10710-010-9112-3

CONTRIBUTED ARTICLE

Human-competitive results produced by genetic programming

John R. Koza

Received: 29 November 2009 / Revised: 17 April 2010 / Published online: 25 May 2010

Abstract Genetic programming has now been used to produce at least 76 instances of results that are competitive with human-produced results. These human-competitive results come from a wide variety of fields, including quantum computing circuits, analog electrical circuits, antennas, mechanical systems, controllers, game playing, finite algebras, photonic systems, image recognition, optical lens systems, mathematical algorithms, cellular automata rules, bioinformatics, sorting networks, robotics, assembly code generation, software repair, scheduling, communication protocols, symbolic regression, reverse engineering, and empirical model discovery. This paper observes that, despite considerable variation in the techniques employed by the various researchers and research groups that produced these human-competitive results, many of the results share several common features. Many of the results were achieved by using a developmental process and by using native representations regularly used by engineers in the fields involved. The best individual in the initial generation of the run of genetic programming often contains only a small number of operative parts. Most of the results that duplicated the functionality of previously issued patents were novel solutions, not infringing solutions. In addition, the production of human-competitive results, as well as the increased intricacy of the results, are broadly correlated to increased availability of computing power tracked by Moore's law. The paper ends by predicting that the increased availability of computing power (through both parallel computing and Moore's law) should result in the production, in the future, of an increasing flow of human-competitive results, as well as more intricate and impressive results.

Keywords Genetic programming · Human-competitive results · Developmental genetic programming · Automated design · Parallel computing · Patented inventions · Moore's law

J. R. Koza (✉)
Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA
e-mail: john@johnkoza.com

1 Introduction

The goal of getting computers to automatically solve problems is central to artificial intelligence, machine learning, and the broad area of research encompassed by what Alan Turing called "machine intelligence" [1, 2].

As early as 1948, Turing recognized the possibility of employing the processes of natural selection and evolution to achieve machine intelligence. In his essay "Intelligent Machines," Turing [1] identified three approaches for creating intelligent computer programs.

The first approach was a logic-driven search. Turing's interest in this approach is not surprising in light of Turing's own pioneering work in the 1930s on the logical foundations of computing.

The second approach for achieving machine intelligence was what Turing called a "cultural search" in which previously acquired knowledge is accumulated, stored in libraries, and brought to bear in solving a problem—the approach taken by subsequent work in the field of knowledge-based expert systems.

The third approach that Turing identified in 1948 for achieving machine intelligence is:

> "… the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value."

In his 1950 paper "Computing Machinery and Intelligence," Turing described how evolution and natural selection might be used to automatically create an intelligent computer program [2].

> "We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications
> "Structure of the child machine = Hereditary material"
> "Changes of the child machine = Mutations"
> "Natural selection = Judgment of the experimenter"

Thus, Turing correctly perceived in 1948 and 1950 that machine intelligence might be achieved by an evolutionary process in which a description of a computer program (the hereditary material) undergoes progressive modification (mutation) under the guidance of natural selection (i.e., selective pressure in the form of what is today usually called "fitness" by practitioners of genetic and evolutionary computation). However, Turing did not envision a population of entities being involved in the search process or any analog of sexual recombination.

In the 1975 book *Adaptation in Natural and Artificial Systems*, John Holland formalized the notion of "combination[s] of genes" and employed a population of entities in a genetic search [3].

Genetic programming is an extension of John Holland's genetic algorithm to the domain of computer programs. Specifically, genetic programming breeds an ever-improving population of programs by iteratively transforming a population of computer programs into a new generation of programs using the Darwinian principle of natural selection and analogs of naturally occurring genetic operations such as recombination (crossover), mutation, gene duplication, gene deletion, and mechanisms of developmental biology [4–15].

The main points of the 1992 book *Genetic Programming: On the Programming of Computers by Means of Natural Selection* [6] were that

- Virtually all problems in artificial intelligence, machine learning, adaptive systems, and automated learning can be recast as a search for a computer program.
- Genetic programming provides a way to successfully conduct the search for the desired computer program in the space of computer programs.

Genetic programming developed from the seminal work of numerous researchers in the 1970s and 1980s. Holland [3] discussed the possibility of using the genetic algorithm to evolve sequences of assembly code. In 1978, Holland also proposed a broadcast language in which the genetic algorithm operated on structures more complex than fixed-length character strings. Holland and Reitman [16, 17] introduced the genetic classifier system in which sets of if–then logical production rules were evolved by means of a genetic algorithm. In 1980, Stephen F. Smith [18] introduced the variable-length genetic algorithm and applied it to populations consisting of a hierarchy of if–then rules. In particular, Smith introduced a crossover operation for populations of if–then rules. In an often-overlooked 1981 paper, Forsyth [19] introduced an innovative system called BEAGLE (Biological Evolutionary Algorithm Generating Logical Expressions) in which logical expressions were evolved in an evolutionary process. In the mid-1980s, Nichael Cramer [20] described innovative experiments in program induction employing Smith's crossover operation [18]. Hicklin (a student of John Dickinson at the University of Idaho) described a system with mutation and reproduction of computer programs [21]; Fujiki (another of Dickinson's students) applied all genetic operations (including a crossover operation) to logical programs [22]; Fujiki and Dickinson performed induction of if–then clauses for playing the iterated prisoner's dilemma game [23]; Antonisse and Keller applied genetic methods to higher-level representations [24]; and Bickel and Bickel applied genetic methods to if–then expert system rules [25].

Since this early research in genetic programming, the field has grown significantly, and work in the field of genetic programming is regularly reported in numerous publications and conferences, including the journal *Genetic Programming and Evolvable Machines*; the journal *IEEE Transactions on Evolutionary Computation*; the annual Genetic and Evolutionary Computation Conference (GECCO) [26] combining the former Genetic Programming Conference held between 1996 and 1998 [27–29] and the former International Conference on Genetic Algorithms held between 1985 and 1997 [30]; the annual Euro-Genetic Programming conference [31]; the annual Genetic Programming Theory and Applications workshop [32]; the Asia–Pacific Workshops on Genetic Programming [33]; edited collections of papers such as the three *Advances in Genetic Programming* books [34–36]; and in various other journals and conferences in the field of genetic and evolutionary computation. In addition, applications of genetic programming appear in publications devoted to the subject matter of the work. Additional sources of information about genetic programming, including links to available software for implementing genetic programming, can be found at www.genetic-programming.org and at the Genetic Programming bibliography at http://www.cs.bham.ac.uk/~wbl/biblio

The aim of the field of machine intelligence is, to paraphrase Arthur Samuel [37] from the 1950's:

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

The criterion for success for the field of machine intelligence is often couched in terms of comparisons with results produced by humans. For example, Samuel [38] stated in 1983,

> "The aim [is] to get machines to exhibit behavior, which if done by humans, would be assumed to involve the use of intelligence."

We say that a result produced by an automated method of machine intelligence is "human-competitive" if it satisfies one of the following eight criteria [13]:

(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 1 shows 76 instances of work (of which the author is currently aware) where genetic programming has produced a result that can be called "human competitive" in accordance with the above criteria. As can be seen, these 76 results come from a wide variety of fields, including quantum computing circuits, analog electrical circuits, antennas, mechanical systems, controllers, game playing, finite algebras, photonic systems, image recognition, optical lens systems, mathematical algorithms, cellular automata rules, bioinformatics, sorting networks, robotics, assembly code generation, software repair, scheduling, communication protocols, symbolic regression, reverse engineering, and empirical model discovery.

Table 1 Human-competitive results produced by genetic programming

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
1	1994	John R. Koza	Creation of algorithm for the transmembrane segment identification problem for proteins			[39–41]; Sects. 18.8 and 18.10 of [8]
2	1995	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Creation of motifs that detect the D–E–A–D box family of proteins and the manganese superoxide dismutase family			[42], Sect. 59.8 of [10]
3	1996	David Andre, Forrest H Bennett III, and John R. Koza	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 at the time			[43], Sect. 58.4 of [10]
4	1996	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Rediscovery of the Campbell ladder topology for lowpass and highpass filters		1917 U.S. patent 1,227,113 by George Campbell	[44, 45], Sect. 25.15.1 of [10], Sect. 5.2 of [13]
5	1996	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Automatic decomposition of the problem of synthesizing a crossover filter		1925 U.S. patent 1,538,964 by Otto Julius Zobel	[46], Sect. 32.3 of [10]
6	1996	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Rediscovery of a recognizable voltage gain stage and a Darlington emitter-follower section of an amplifier and other circuits		1953 U.S. patent 2,663,806 by Sidney Darlington	[47], Sect. 42.3 of [10]
7	1996	John R. Koza, David Andre, Forrest H Bennett III, and Martin A. Keane	Automatic synthesis of 60 dB and 96 dB amplifiers		Numerous patents	[48, 49], Sect. 45.3 of [10]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
8	1996	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Automatic synthesis of asymmetric bandpass filter			[46]
9	1997	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Creation of a sorting network for seven items using only 16 steps		1962 U.S. patent 3,029,413 by Daniel G. O'Connor and Raymond J. Nelson	[50–52], Sects. 21.4.4, 23.6, and 57.8.1 of [10]
10	1997	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Automatic synthesis of a real-time analog circuit for time-optimal control of a robot			[53] Chapter 48 of [10]
11	1997	John R. Koza, Forrest H Bennett III, Martin A. Keane, and David Andre	Automatic synthesis of analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions		Numerous patents	[54], Sect. 47.5.3 of [10]
12	1998	Lee Spector, Howard Barnum, and Herbert J. Bernstein	Creation of a better-than-classical quantum algorithm for the Deutsch-Jozsa "early promise" problem			[55]
13	1998	Lee Spector, Howard Barnum, and Herbert J. Bernstein	Creation of a better-than-classical quantum algorithm for Grover's database search problem			[56]
14	1998	Sean Luke	Creation of a soccer-playing program that won its first two games in the Robo Cup 1997 competition			[57]
15	1999	David Andre and Astro Teller	Creation of a soccer-playing program that ranked in the middle of the field of 34 human-written programs in the Robo Cup 1998 competition			[58]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
16	1999	Lee Spector, Howard Barnum, and Herbert J. Bernstein, and N. Swamy	Creation of a quantum algorithm for the depth-two AND/OR query problem that is better than any previously published result			[59, 60]
17	1999	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Creation of four different algorithms for the transmembrane segment identification problem for proteins			Sections 16.5 and 17.2 of [10]
18	1999	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Rediscovery of "*M*-derived half section" and "constant *K*" filter sections		1925 U.S. patent 1,538,964 by Otto Julius Zobel	Section 25.15.3 of [10]
19	1999	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Rediscovery of the Cauer (elliptic) topology for filters		1934 and 1936 U.S. patents 1,958,742 and 1,989,545 by Wilhelm Cauer	Section 27.3.7 of [10]
20	1999	Forrest H Bennett III, John R. Koza, Martin A. Keane, Jessen Yu, William Mydlowec, and Oscar Stiffelman	Automatic synthesis of digital-to-analog converter (DAC) circuit			[61]
21	1999	Forrest H Bennett III, John R. Koza, Martin A. Keane, Jessen Yu, William Mydlowec, and Oscar Stiffelman	Automatic synthesis of analog-to-digital (ADC) circuit			[61]
22	1999	Forrest H Bennett III, John R. Koza, Martin A. Keane, Jessen Yu, William Mydlowec, and Oscar Stiffelman	Automatic synthesis of a NAND circuit		1971 U.S. patent 3,560,760 by David H. Chung and Bill H. Terrell	[61], Sect. 4.4 of [13]
22	1999	Forrest H Bennett III and John R. Koza,	Automatic synthesis of topology, sizing, placement, and routing of analog electrical circuits			[62]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
23	1999	John R. Koza, Forrest H Bennett III, Martin A. Keane, Jessen Yu, William Mydlowec, and Oscar Stiffelman	Automatic synthesis of analog circuit equivalent to Philbrick circuit		1956 U.S. patent 2,730,679 by George Philbrick	Section 4.3 of [13], [63]
24	1999	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Automatic synthesis of an electronic thermometer		Numerous patents	Section 49.3 of [10]
25	1999	John R. Koza, Forrest H Bennett III, David Andre, and Martin A. Keane	Automatic synthesis of a voltage reference circuit		Numerous patents	Section 50.3 of [10]
26	2000	John R. Koza, Martin A. Keane, Yu, Jessen, Forrest H Bennett III, and William Mydlowec	Automatic synthesis of PID (proportional, integrative, and derivative) controller		1939 U.S. patent 2,175,985 by Albert Callender and Allan Stevenson	[64], Sect. 9.2 of [13]
27	2000	John R. Koza, Martin A. Keane, Yu, Jessen, Forrest H Bennett III, and William Mydlowec	Automatic synthesis of a PID-D2 (second derivative) type of controller		1942 U.S. patent 2,282,726 by Harry Jones	[64], Sect. 9.2 of [13]
28	2000	Howard Barnum, Herbert J. Bernstein, and Lee Spector	Creation of a quantum algorithm for the depth-one OR query problem that is better than any previously published result			[60]
29	2002	Martin A. Keane, John R. Koza, and Matthew J. Streeter	Creation of PID tuning rules that outperform the Ziegler-Nichols and Åström-Hägglund tuning rules		2005 U.S. patent 6,847,851 by Martin A. Keane, John R. Koza, and Matthew J. Streeter	[65, 66], chapter 12 of [13]
30	2002	Martin A. Keane, John R. Koza, and Matthew J. Streeter	Creation of three non-PID controllers that outperform a PID controller using the Ziegler-Nichols or Åström-Hägglund tuning rules		2005 U.S. patent 6,847,851 by Martin A. Keane, John R. Koza, and Matthew J. Streeter	Chapter 13 of [13], [66]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
31	2002	Matthew J. Streeter, Martin A. Keane, and John R. Koza	Cubic function generator		2000 U.S. patent 6,160,427 by Stefano Cipriani and Anthony A. Takeshian	Section 15.4.5 of [13], [67]
32	2002	Matthew J. Streeter, Martin A. Keane, and John R. Koza	Mixed analog–digital variable capacitor circuit		2000 U.S. patent 6,013,958 by Turgut Sefket Aytur	Section 15.4.2 of [13], [67]
33	2002	Matthew J. Streeter, Martin A. Keane, and John R. Koza	Voltage-current conversion circuit		2000 U.S. patent 6,166,529 by Akira Ikeuchi and Naoshi Tokuda	Section 15.4.4 of [13], [67]
34	2002	Matthew J. Streeter, Martin A. Keane, and John R. Koza	Low-voltage balun circuit		2001 U.S. patent 6,265,908 by Sang Gug Lee	Section 15.4.1 of [13], [67]
35	2002	Matthew J. Streeter, Martin A. Keane, and John R. Koza	High-current load circuit		2001 U.S. patent 6,211,726 by Timothy Daun-Lindberg and Michael Miller	Section 15.4.3 of [13], [67]
36	2003	John R. Koza, Martin A. Keane, Matthew J. Streeter, William Mydlowec, Jessen Yu, and Guido Lanza	Tunable integrated active filter		2001 U.S. patent 6,225,859 by Robert Irvine and Bernd Kolb	Section 15.4.6 of [13]
37	2003	Lee Spector and Herbert J. Bernstein	Creation of a protocol for communicating information through a quantum gate that was previously thought not to permit such communication			[68]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
38	2003	Lee Spector and Herbert J. Bernstein	Creation of a novel variant of quantum dense coding			[68]
39	2003	John R. Koza, Martin A. Keane, Matthew J. Streeter, William Mydlowec, Jessen Yu, and Guido Lanza	Negative feedback		1937 U.S. patents 2,102,670 and 2,102,671 by Harold S. Black	Chapter 14 of [13]
40	2004	Jason D. Lohn, Gregory S. Hornby, and Derek S. Linden	An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission	Gold		[69, 70]
41	2004	Lee Spector	*Automatic Quantum Computer Programming: A Genetic Programming Approach*	Gold		[71, 60, 68]
42	2004	Alex Fukunaga	Evolving Local Search Heuristics for SAT Using Genetic Programming; Automated discovery of composite SAT variable-selection heuristics	Silver		[72, 73]
43	2004	Hod Lipson	Mechanical system composed of rigid members for drawing a straight line; How to Draw a Straight Line Using a GP: Benchmarking Evolutionary Design Against 19th Century Kinematic Synthesis	Silver	1841 Great Britain patent 6258 by Robert Willis	[74, 75]
44	2004	Bijan KHosraviani, Raymond E. Levitt, and John R. Koza	Organization Design Optimization Using Genetic Programming	Silver		[76]
45	2004	Brian Lam	Discovery of Human-Competitive Image Texture Feature Programs Using Genetic programming	Merit		[77]
46	2004	Lukas Sekanina	Novel image filters implemented in hardware	Merit		[78]
47	2005	Stefan Preble; Hod Lipson; Michal Lipson	Two-dimensional photonic crystals designed by evolutionary algorithms	Gold		[79]
48	2005	John Koza; Sameer Al-Sakran; Lee Jones	Telescope eyepiece; Automated Re-Invention of Six Patented Optical Lens Systems using Genetic Programming	Silver	1940 U.S. patent 2,206,195 by Albert Konig	[80, 81]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
49	2005	John Koza; Sameer Al-Sakran; Lee Jones	Telescope eyepiece system	Silver	1958 U.S. patent 2,829,560 by Robert B. Tackaberry and Robert M. Muller	[80, 81]
50	2005	John Koza; Sameer Al-Sakran; Lee Jones	Eyepiece for optical instruments	Silver	1953 U.S. patent 2,637,245 by Maximillian Ludewig	[80, 81]
51	2005	John Koza; Sameer Al-Sakran; Lee Jones	Wide angle eyepiece	Silver	1968 U.S. patent 3,390,935 by Wright H Scidmore	[80, 81]
52	2005	John Koza; Sameer Al-Sakran; Lee Jones	Wide angle eyepiece	Silver	1985 U.S. patent 4,525,035 by Albert Nagler	[80, 81]
53	2005	John Koza; Sameer Al-Sakran; Lee Jones	Telescope eyepiece	Silver	2000 U.S. patent 6,069,750 by Noboru Koizumi and Naomi Watanabe	[80, 81]
54	2005	Paul Massey; John A Clark; Susan Stepney	Evolution of a Human-Competitive Quantum Fourier Transform Algorithm Using Genetic Programming	Silver		[82]
55	2005	Fulvio Corno; Edgar Ernesto Sanchez Sanchez; Giovanni Squillero	Evolving Assembly Programs: How Games Help Microprocessor Validation	Silver		[83]
56	2005	Moshe Sipper; Yaniv Azaria; Ami Hauptman; Yoanatan Shichel; Eran Ziserman	Attaining Human-Competitive Game Playing with Genetic Programming; GP-Gammon: Using Genetic Programming to Evolve Backgammon Players; GP-Gammon: Genetically Programming Backgammon Players	Bronze		[84–86]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
57	2005	Moshe Sipper; Yaniv Azaria; Ami Hauptman; Yoanatan Shichel; Eran Ziserman	GP-EndChess: Using Genetic Programming to Evolve Chess Endgame	Bronze		[84, 87]
58	2005	Moshe Sipper; Yaniv Azaria; Ami Hauptman; Yoanatan Shichel; Eran Ziserman	GP-Robocode: Using Genetic Programming to Evolve Robocode Players	Bronze		[84, 88]
59	2005	Joc Cing Tay; Nhu Binh Ho	Evolving Dispatching Rules for Solving the Flexible Job-Shop Problem	Honorable Mention		[89]
60	2005	Daniel Howard; Joseph Kolibal	Solution of differential equations with Genetic Programming and the Stochastic Bernstein Interpolation	Honorable Mention		[90]
61	2005	Maarten Keijzer; Martin Baptist; Vladan Babovic; Javier Uthurburu	Determining Equations for Vegetation Induced Resistance using Genetic Programming; Modeling Floodplain Biogeomorphology Resistance using Genetic Programming	Honorable Mention		[91, 92]
62	2005	John R. Koza, Martin A. Keane, Matthew J. Streeter, and Thomas Adams	Sallen-Key filter			[93]
53	2006	Gustavo Olague; Leonardo Trujillo	Using Evolution to Learn How to Perform Interest Point Detection; Synthesis of Interest Point Detectors Through Genetic Programming	Bronze		[94, 95]
64	2007	Ami Hauptman; Moshe Sipper	Evolution of an Efficient Search Algorithm for the Mate-In-N Problem in Chess	Silver		[96]
65	2007	Ronan Cummins; Colm O’Riordan	Evolving local and global weighting schemes in Information Retrieval; An analysis of the Solution Space for Genetically Programmed Term-Weighting Schemes in Information Retrieval; Term-Weighting in Information Retrieval using Genetic Programming: A Three-Stage Process	Honorable mention		[97–99]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
66	2007	Muhammad Adil Raja; Raja Muhammad Atif Azad; Colin Flanagan; Conor Ryan	Real-Time, Non-Intrusive Evaluation of VoIP	Honorable mention		[100]
67	2007	Josh Bongard; Hod Lipson	Automated Reverse Engineering of nonlinear Dynamical Systems			[101]
68	2007	Amr Radi; Salah Yaseen El-Bakry; Mostafa Y. El-Bakry	Genetic programming approach for electron-alkali-metal atom collisions; Prediction of Non Linear System in Optics Using Genetic Programming; Genetic programming approach for flow of steady state fluid between two eccentric spheres			[102–104]
69	2008	Lee Spector, David M. Clark, Ian Lindsay, Bradford Barr, Jon Klein	Genetic Programming for Finite Algebras	Gold		[105]
70	2008	Lee Spector and Jon Klein	Automatic synthesis of quantum computing circuit for the two-oracle AND/OR problem			[106]
71	2008	Ralf Stadelhofer, Wolfgang Banzhaf, and Dieter Suter	Automatic synthesis of quantum computing algorithms for the parity problem a special case; of the hidden subgroup problem			[107]
72	2008	Jianjun Hu, Erik D. Goodman, Shaobo Li, and Ronald Rosenberg	Automatic synthesis of mechanical vibration absorbers		1911 U.S. patent 989,958 by H. Frahm	[108]
73	2009	Stephanie Forrest, Claire Le Goues, ThanhVu Nguyen, and Westley Weimer	Automatically finding patches and automated software repair	Gold		[109, 110]
74	2009	Ami Hauptman; Achiya Elyasaf; Moshe Sipper; Assaf Karmon	GP-Rush: Using Genetic Programming to Evolve Solvers for the Rush Hour Puzzle	Bronze		[111]

Table 1 continued

	Year	Authors	Title	Human-Competitive Prize Award	Patent	References
75	2009	Cynthia B. Perez; Gustavo Olague	Learning Invariant Region Descriptor Operators with Genetic Programming and the F-measure; Evolutionary Learning of Local Descriptor Operators for Object Recognition	Bronze		[112, 113]
76	2009	P. Balasubramaniam, and A. Vincent Antony Kumar.	Solution of matrix Riccati differential equation for nonlinear singular system using genetic programming			[114]
77	2009	Michael Schmidt and Hod Lipson	Distilling Free-Form Natural Laws from Experimental Data			[115, 116]

In Table 1, there are 31 instances (shown in column 5) where the human-competitive result was recognized by the annual Human Competitive Awards program (www.human-competitive.org) started in 2004 at the Genetic and Evolutionary Computation Conference (GECCO). Note that there are 77 items the table because the prize award recognized by item 41 is subsumed by other items in the table.

In addition, there are 31 instances (shown in column 6 of Table 1) where the human-competitive result produced by genetic programming duplicated the functionality of a previously patented invention, infringed a previously issued patent, or created a patentable new invention. These include one instance where genetic programming has created an entity that either infringes or duplicates the functionality of a previously patented 19th-century invention, 21 instances where genetic programming has done the same with respect to previously patented 20th-century inventions, 7 instances where genetic programming has done the same with respect to previously patented 21st-century inventions, and two instances where genetic programming has created a patentable new invention [13, 66]. The first invention of these patentable new inventions covers improved tuning rules for PID (proportional, integrative, and derivative) controllers. The second is for general-purpose (non-PID) controllers that outperform controllers based on the widely used Ziegler-Nichols tuning rules [117] and the Åström-Hägglund tuning rules [118].

Despite considerable variation in representation and techniques employed by the various research groups that produced the work shown in Table 1, many of these human-competitive results shared several common features.

- Developmental genetic programming was used to produce many of the results (Sect. 2).
- Native representations were used to produce many of the results (Sect. 3).
- The best individual in early generations often contains only a very small number of operative parts (Sect. 4).
- Most of the results that duplicated the functionality of previously issued patents were novel solutions (Sect. 5).
- The production of human-competitive results, as well as the increased intricacy of the results, are broadly correlated to increased availability of computing power suggested by Moore's law (Sect. 6).

Section 7 discusses future possibilities for producing additional human-competitive results using genetic programming in line with the increased availability of computing power tracked by Moore's law and made possible through parallel computing.

2 Developmental genetic programming

Many of the human-competitive results in Table 1 were produced using runs of genetic programming that employed a developmental process. Examples include work on design of

- quantum computing circuits by Spector [71] and his colleagues,
- antennas by Lohn, Hornby, and Linden [69, 70],

- mechanical systems by Lipson [74, 75],
- analog electrical circuits by Koza, Bennett, Andre, Keane, Streeter, Mydlowec, Yu, and Lanza [10, 13],
- photonic crystals by Preble, Lipson, Lipson [79],
- optical lens systems by Koza, Al-Sakran, and Jones [80, 81], and
- negative feedback by Koza, Keane, Streeter, Mydlowec, Yu, and Lanza [13].

When developmental genetic programming is used, the individuals that are genetically bred during the run of genetic programming are not themselves candidate structures in the domain of interest. Instead, the individuals are computer programs consisting of instructions for constructing the candidate structures. In the developmental approach, the programs in the population, when executed, transform some very simple initial structure (called the *embryo*) into a fully developed structure in the domain of interest. For example, when developmental genetic programming is used to automatically design electrical or quantum computing circuits, the individuals in the population are not circuits, but, instead, computer programs that specify how to construct a circuit, step by step, from some simple initial structure (often just a single wire).

The developmental representations used to apply genetic programming to produce the human-competitive results in Table 1 arise from early work in the field of genetic algorithms and genetic programming. In 1987, Wilson [119] stated:

> "The genetic algorithm observes the genotype-phenotype distinction of biology: the algorithm's variation operators act on the genotype and its selection mechanisms apply to the phenotype. In biology, the genotype-phenotype difference is vast: the genotype is embodied in the chromosomes whereas the phenotype is the whole organism that expresses the chromosomal information. The complex decoding process that leads from one to the other is called biological development and is essential if the genotype is to be evaluated by the environment. Thus to apply the genetic algorithm to natural evolution calls for a representational scheme that both permits application of the algorithm's operators to the genotype and also defines how, based on the genotype, organisms are to be 'grown,' i.e., their development."

Kitano used a developmental process in conjunction with genetic algorithms to design neural networks using a graph generation system in 1990 [120].

In 1992, Gruau [121] described a technique in which genetic programming is used to concurrently evolve the architecture of a neural network, along with the weights, thresholds, and biases of each neuron in the neural network. In Gruau's *Cellular Encoding of Genetic Neural Networks*, each individual program tree in the population of the run of genetic programming is a program for developing a complete neural network from a starting point consisting of a single embryonic neuron. In Gruau's developmental genetic programming approach, the developmental process for a neural network starts from an embryonic neural network consisting of a single neuron. The functions in the program tree specify how to develop the embryonic neural network into a full neural network. Certain functions permit a particular neuron to be subdivided in a parallel or sequential manner. Other functions can change the threshold of a neuron, the weight of a connection, or a neuron's bias. Genetic programming is then used to breed populations of network-constructing program trees in order to evolve a neural network that is capable of solving a particular problem. Gruau also described a version of his system using recursion [122–126]. Whitley, Gruau, and Preatt [127] applied developmental genetic programming to neurocontrol problems.

In 1993, Koza [128] used genetic programming to evolve developmental rewrite rules (Lindenmayer system rules). In this work, a "turtle" moved and created desired patterns, such as the quadratic Koch island.

In 1994, Dellaert and Beer [129] described "the synthesis of autonomous agents using evolutionary techniques" and presented "a simplified yet biologically defensible model of the developmental process."

In 1994, Hemmi, Mizoguchi, and Shimohara [130] noted, "Using a rewriting system, the system introduces a program development process that imitates the natural development process from the pollinated egg to adult and gives the HDL-program flexible evolvability."

In 1994, Sims [131] describes a system in which the morphological and behavioral components of virtual creatures are represented by directed graphs that evolve through the use of a graph-based genetic algorithm.

In 1996, Koza, Bennett, Andre, and Keane used developmental genetic programming to automatically synthesize a large body of analog electrical circuits, including several previously patented circuits [44, 46, 47, 132]. Circuit-constructing functions in the program tree specified how to develop a simple embryonic circuit (often containing just a single modifiable wire) into a fully developed circuit (containing transistors, capacitors, resistors, and other electronic components). Their method permitted the construction of a via to connect distant parts of the circuit. They also provided for reuse of portions of circuits (by means of subroutines and iterations), parameterized reuse, and hierarchical reuse of substructures in evolving circuits [133].

In 1996, Brave [134] used developmental genetic programming to evolve finite automata.

In 1996, Tunstel and Jamshidi used developmental methods for fuzzy logic systems [135].

In 1996, Spector and Stoffel [136, 137] extended the notion of development to what they called "ontogenetic programming."

> "In nature, the structure and behavior of a mature organism is determined not only by its genetic endowment, but also by complex developmental processes that the organism undergoes while immersed in its environment (ontogeny)."
> …
> "Biologists refer to the developmental progression of an individual through its life span as ontogeny. In this paper we describe how rich ontogenetic components can be added to genetic programming systems, and we show how this can allow genetic programming to produce programs that solve more difficult problems." …

> "Various morphological systems have been used in previous genetic programming systems to allow programs to 'grow' into more complex forms prior to evaluation. Runtime memory mechanisms allow evolved programs to acquire information from their environments while they solve problems, and to change their future behavior on the basis of such information."
> "Ontogenetic programming combines these ideas to allow for runtime modification of program structure. In particular, an ontogenetic programming system includes program self-modification functions in the genetic programming function set, thereby allowing evolved programs to modify themselves during the course of the run." …
> "[W]e show how ontogenetic programming can be used to solve problems that would otherwise not be solvable." …
> "We have shown that it is possible to use genetic programming to produce programs that themselves develop in significant, structural ways over the course of a run. We use the term 'ontogenetic programming' to describe our technique for achieving this effect, which involves the inclusion of program self-modification functions in the genetic programming function set." [137]

In 1996, Spector and Stoffel applied their methods to a symbolic regression problem [136], a sequence prediction problem [136], and a robotic agents problem [137]. They also describe how their methods can be used in conjunction with both tree and linear representations [137].

In 1996, Luke and Spector [138] describe yet another variation on the developmental process:

> "Like a cellular encoding, an edge encoding is a tree-structured chromosome whose phenotype is a directed graph, optionally with labels or functions associated with its edges and nodes. …
> "Edge encoding, like cellular encoding, allows one to use standard S-expression-based Genetic Programming techniques to evolve arbitrary graph structures. The resulting graphs may be employed in various ways, for example as neural networks, as automata, or as knowledge-base queries. Each encoding scheme biases genetic search in a different way; for example, cellular encoding favors graphs with high edge/node ratios while edge encoding favors graphs with low edge/node ratios. For this reason, we believe that certain domains will be much better served by one scheme than by the other."

3 Native representations

It would be reasonable to think that a human employing genetic programming would need considerable insight and in-depth knowledge in order to identify a representational scheme (that is, the function set and terminal set) that can yield human-competitive results in a particular field. Thus, it is noteworthy that many of the human-competitive results in Table 1 were produced merely by using the "native" representational structures that are regularly used by engineers in the field involved. Examples of such native representations include the work in Table 1 on the automatic synthesis of

- optical lens systems by Koza, Al-Sakran, and Jones [80, 81], and
- analog electrical circuits by Koza, Bennett, Andre, Keane, Streeter, Mydlowec, Yu, and Lanza [10, 13],
- antennas by Lohn, Hornby, and Linden [69, 70],
- mechanical systems by Lipson [74, 75], and
- quantum computing circuits by Spector [71] and his colleagues.

For example, for optical lens systems [80, 81], the representational scheme used to create human-competitive designs in Table 1 was simply the format for an optical *prescription* that is regularly used in the field of optical design. Interestingly, the prescription also corresponds directly to the "lens file" used by many pieces of software for the simulation of optical lens systems. The function set for problems of spherical refractive optics consists of one key function that inserts a material (e.g., a specified type of glass, air, oil, vacuum) and a surface (with a specified radius of curvature) at a specified distance from the previous surface (or starting point). This function corresponds to the information contained in consecutive rows of a standard optical prescription (or lens file). No knowledge of the mathematics for analyzing the behavior of optical system (e.g., laws of optics, ray tracing, modulation transfer functions, spot diagrams) or practical engineering know-how or domain-specific rules-of-thumb about optical design was involved in arriving at this "native" representational scheme. Instead, all that was used was syntactic information about one suitable (and widely used) way to unambiguously describe the structure of a lens system.

The situation was similar for analog electrical circuits [10, 13]. In the field of analog circuits, the function and terminal set consists of functions that insert electrical components (e.g., transistors, capacitors, resistors, inductors) and create topological connections among the components in well-known ways (e.g., parallel division, series division, a via connecting arbitrary points). The resulting circuit can be represented by a tabular "netlist" in which each row describes an electrical component and identifies the component's connections to other components (or input port, output ports, power sources, and ground points). This "netlist" data structure is then often the primary input to software for circuit simulation.

Similarly, for wire antennas [69, 70, 139], the representational scheme is based on the way a draftsman might draw an antenna (either on paper or using a two- or three-dimensional graphical interface for engineering drawings). In particular, the function set consists of one key function that inserts a straight segment of metal wire with a specified length (or no wire) and at a specified angle starting at a specified point, using a turtle [128, 139]. The resulting antenna can be represented by a tabular "geometry table," and this data structure is then often the primary input to software for antenna simulation.

For mechanical systems [74, 75], the function set consists of two functions that construct the mechanical device while maintaining the originally specified degrees-of-freedom of the embryonic starting structure. One function modifies a link by attaching two new linked nodes to each end of the specified link, with two numerical arguments specifying the directional orientation of the to-be-added triangle. A second function modifies a link by replacing it with two new links that pass through a newly created common node, with the two numerical arguments specifying the direction of growth (the free end of each of the new links being attached to the mechanism at the closest nodes).

For quantum computing circuits [71], the function set consists of quantum gates that modify the state of the evolved systems qubits. These include a function that inverts the states of a qubit, a function that inverts the state of a qubit conditional on another qubit's value, a function that rotates the qubit, a function that swaps the state of two qubits, as well as the square root of a NOT gate, the Hadamard Gate, a generalized rotation function, a controlled phase function, and a function that allows the state of a qubit to be read.

For controllers [13, 66], the function set consists of the signal-processing functions that are commonly used in controllers (e.g., integrators, differentiators, amplifiers, adders, subtractors, leads, lags). A controller is simply a composition of these signal-processing functions (i.e., a LISP S-expression) operating on the controller's two externally supplied signals (i.e., the reference signal and the plant output).

Of course, the native representations chosen by the various research groups that performed the above work are not necessarily the most efficient representations. Nonetheless, it is noteworthy that the various research groups involved were each able to generate human-competitive results using genetic programming by using a straight-forward representational scheme employing only basic information about the field involved. This fact suggests that genetic programming may prove able to readily yield additional human-competitive results when it is applied to design problems in many other fields.

The fact that native representations can be successfully used by the evolutionary process has the secondary advantage of making the results readable and understandable to human engineers (thereby enhancing trust in the results).

4 The best individual in early generations often contains only a very small number of operative parts

The initial population (generation 0) for a run of genetic programming is typically created using a probabilistic growth process for program trees (or other non-tree representations that might be used). Generally, this growth process is structured so as to guarantee the creation of a diverse population of individuals having a wide range of sizes and shapes.

There is a loose and imperfect correlation between program size and the number of operative parts in the structure. One of the common features of runs of genetic programming that produced human-competitive result is that the best-of-generation structure of generation 0 tends to have a small number of operative parts.

For example, consider recent work in which genetic programming was used to automatically synthesize complete design for the Konig optical system patented in 1940 [80]. The best individual of generation 0 (Fig. 1a) is a lens system with one positive lens of flint glass. This single lens manages to perform the important threshold task of focusing the axial rays (a, b, and c in the figure) coming in through the entry pupil onto (very approximately) the desired area of the image surface. Of course, there is more to optical design than merely focusing light rays on an image surface. This single lens does nothing in terms of satisfying the multiplicity of additional specifications contained in the Konig's patent. However, this individual provides a toehold from which the evolutionary process can make further progress. For comparison, Fig. 1b shows the best-of-run lens optical lens system for a successful run. This four-lens system satisfies the design requirements specified for the run and infringes on the Konig (1940) patent. As can be seen, the final circuit has considerably more operative parts than the best-of-generation individual from generation 0.

Turning now to the field of the automated synthesis of analog electrical circuits, Fig. 2a shows the best-of-generation circuit from generation 0 of a run in which the goal was to synthesize the topology and sizing for a lowpass filter with requirements on passband ripple and stopband attenuation equivalent to that of a fifth-order elliptic filter [10, 44, 45]. As can be seen, this circuit contains only one capacitor shunting the input signal to ground. Nonetheless, even a single shunt capacitor filters an incoming signal to some degree. Figure 2b shows the best-of-run circuit for the lowpass filter problem. This circuit consists of a cascade of seven rungs of a ladder, with each rung of the ladder consisting of a shunt capacitor (such as appeared in generation 0) and a series inductor. This circuit satisfies the design requirements of the problem and infringes the 1917 patent of George Campbell.

Turning to the field of the automated synthesis of controllers, Fig. 3a shows the best-of-generation controller from generation 0 for the problem of synthesizing a controller for a two-lag plant problem. The output of this controller is a numerical multiple of the result of subtracting the plant output from a lagged reference signal. This individual does not even remotely satisfy the problem's requirements; however, it does have threshold characteristics of considering the difference between the (lagged) reference signal and the plant output signal and then acting on the difference of these two signals. The best-of-run controller (Fig. 3b) infringes on Jones's 1942 patent.

The reason that best individuals at the beginning of a run of genetic programming tend to have only one (or just a few) operative parts is that, when there is a large number of operative parts, the parts must be appropriately connected and each of the parts must possess numerical parameters that are appropriately coordinated the

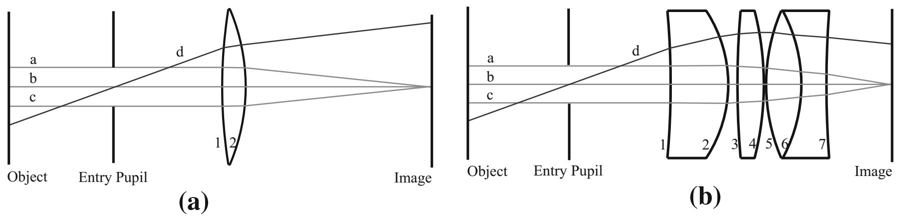

Fig. 1 Comparison of best of generation 0 and best-of-run individuals for the patented Konig optical lens system (a) Best-of-generation 0 (b) Best-of-run

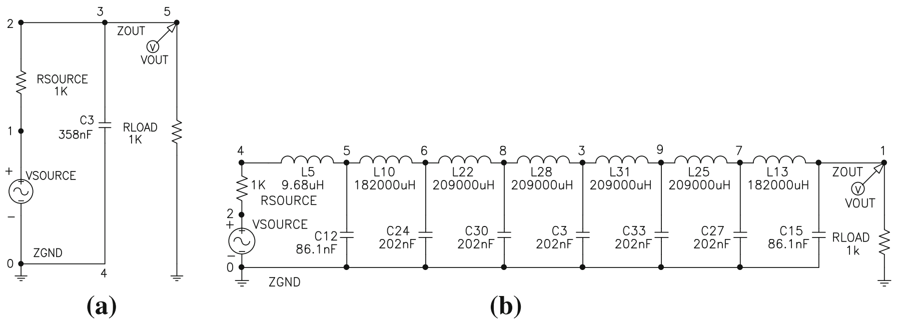

Fig. 2 Comparison of best of generation 0 and best-of-run individuals for the patented Campbell filter (a) Best-of-generation 0 (b) Best-of-run

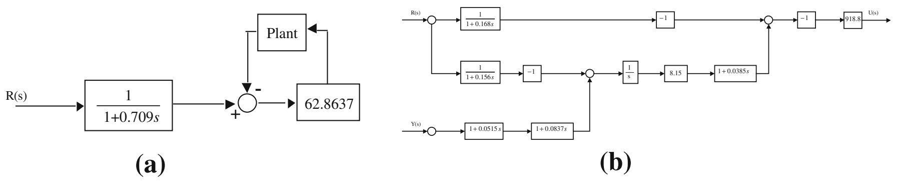

Fig. 3 Comparison of best of generation 0 and best-of-run individuals for the patented Jones controller (a) Best-of-generation 0 (b) Best-of-run

numerical parameters of all the other parts. As the number of operative parts increases, the probability becomes diminishingly small that the random growth process used to create generation 0 of the genetic population contains a multi-part structure with reasonably high fitness.

For example, if a candidate filter circuit at the beginning of a run of genetic programming had multiple non-degenerate components (say, two capacitors and two inductors that actively affected the circuit's output), then both capacitors would (probably) have to be positioned so as to shunt the incoming signal down to ground and both inductors would (probably) have to be positioned in series between the incoming signal and the circuit's output point. Moreover, the numerical values of both the capacitors and both of the inductors would have to be suitably coordinated in order to produce anything that resembled a filter.

5 Novelty of results involving previously patented inventions

Table 1 contains two groups of six previously patented inventions that permits us to make the observation that most of the results that duplicated the functionality of previously issued patents did not infringe the previously issued patent, but, instead, solved the problem involved in a novel way.

The first group involved complete designs (i.e., both topology and numerical parameters) for six analog electrical circuits that duplicated the functionality of previously patented twenty-first century circuits [13, 67].

- 2000 U.S. 6,160,427 patent by Stefano Cipriani and Anthony A. Takeshian,
- 2000 U.S. 6,013,958 patent by Turgut Sefket Aytur,
- 2000 U.S. 6,166,529 patent by Akira Ikeuchi and Naoshi Tokuda,
- 2001 U.S. 6,265,908 patent by Sang Gug Lee,
- 2001 U.S. 6,211,726 patent by Timothy Daun-Lindberg and Michael Miller, and
- 2001 U.S. 6,225,859 patent by Robert Irvine and Bernd Kolb.

The second group involved complete designs for six optical lens systems that duplicated the functionality of patented lens systems [80, 81].

- 1940 U.S. patent 2,206,195 by Albert Konig,
- 1958 U.S. patent 2,829,560 by Robert B. Tackaberry and Robert M. Muller,
- 1953 U.S. patent 2,637,245 by Maximillian Ludewig,
- 1968 U.S. patent 3,390,935 by Wright H Scidmore,
- 1985 U.S. patent 4,525,035 by Albert Nagler, and
- 2000 U.S. patent 6,069,750 by Noboru Koizumi and Naomi Watanabe.

In both of these two groups of six patents, only one of the human-competitive results produced by genetic programming infringed one of the pre-existing patents. Specifically, genetic programming created a circuit that infringed the 2001 patent of Irvine and Kolb and created an optical lens system that infringed the 1940 Konig patent.

In the other five cases in each group, genetic programming did one of two things. In some cases, it rediscovered the essence of the invention, but did so with an overall structure that did not actually infringe the patent. That is, genetic programming creatively "engineered" around the actual wording of the original patent while using the scientific principle underlying the original patent. In other cases, genetic programming created a novel solution (that is, a new invention) that satisfied the high-level specifications of the original patented invention, but that did not resemble the solution created by the original human inventor(s).

We suggest that the reason for this one-in-six ratio is that there are many ways to satisfy a given set of engineering specifications. A patent is, in general, merely *a* solution for a problem, but not the *only* solution. In addition, a patented approach is not necessarily an optimal solution. Genetic programming conducts a search for a satisfactory solution, but has no *a priori* knowledge about (and hence no particular preference for) the solution that a human may have thought of in the past. Instead, genetic programming seeks a solution that optimizes the fitness measure provided by the human user. That fitness measure does not, in general, encompass all the considerations (conscious or unconscious) that may have passed through the mind of the original human inventor(s).

6 The role of increased availability of computing power tracked by Moore's law

The production of human-competitive results as well as the increased intricacy of the results are broadly correlated to increased availability of computing power tracked by Moore's law.

Table 2 Human-competitive results produced by genetic programming with five computer systems

System	Period	Petacycles (10^{15} cycles) per day for system	Speed-up over previous row in this table	Speed-up over first system in this table	Number of human-competitive results
Serial Texas Instruments LISP machine	1987–1994	0.00216	1 (base)	1 (base)	0
64-node Transtech transputer parallel machine	1994–1997	0.02	9	9	2
64-node Parsytec parallel machine	1995–2000	0.44	22	204	12
70-node Alpha parallel machine	1999–2001	3.2	7.3	1,481	2
1,000-node Pentium II parallel machine	2000–2002	30.0	9.4	13,900	12

The production of human-competitive results using genetic programming has been greatly facilitated by the fact that genetic algorithms and other methods of evolutionary computation can be readily and efficiently parallelized. Many of the results listed in Table 1 employed parallel computing techniques, including (but not limited to) results in the fields of the automatic synthesis of quantum computing circuits, analog electrical circuits, antenna, optical lens systems, and mechanical systems.

Additionally, the production of human-competitive results using genetic programming has facilitated to an even greater degree by the increased availability of computing power, over a period of time, as tracked by Moore's law. Indeed, over the past two decades, the number and level of intricacy of the human-competitive results has progressively grown.

In discussing this progressive growth, we first acknowledge the practical difficulties of trying to precisely compare the amount of computer time required to produce a certain result when that result is produced by different research groups, using different computers (containing chips with different architectures), different operating systems, different communication architectures between the parts of a parallel computing system, different genetic programming software, and somewhat different implementations of the genetic programming algorithm. Moreover, the relationship between computer cycles and the effective amount of computation performed varies considerably from machine to machine. Finally, for most of the items in Table 1, the majority of the computer time used to produce the various results is consumed by simulators peculiar to the different problem domain involved (as opposed to computer time consumed by the genetic programming algorithm).

Having made all of the above disclaimers, there is, nonetheless, data indicating that the production of human-competitive results using genetic programming is broadly correlated with the increased availability of computer power, from year to year, as tracked by Moore's Law.

Table 3 Progression of qualitatively more substantial results produced by genetic programming in relation to five order-of-magnitude increases in computational power

System	Period	Speed-up	Qualitative nature of results produced by genetic programming
Serial Texas Instruments LISP machine	1987–1994	1 (base)	Toy problems of the 1980s and early 1990s from the fields of artificial intelligence and machine learning [6, 8]
64-node Transtech transputer parallel machine	1994–1997	9	Two human-competitive results involving one-dimensional discrete data (not patent-related) [10]
64-node Parsytec parallel machine	1995–2000	22	One human-competitive result involving two-dimensional discrete data [10]
			Numerous human-competitive results involving continuous signals analyzed in the frequency domain [10]
			Numerous human-competitive results involving 20th-century patented inventions [10]
70-node Alpha parallel machine	1999–2001	7.3	One human-competitive result involving continuous signals analyzed in the time domain [13]
			Circuit synthesis extended from topology and sizing to include routing and placement (layout) [13]
1,000-node Pentium II parallel machine	2000–2002	9.4	Numerous human-competitive results involving continuous signals analyzed in the time domain [13]
			Numerous general solutions to problems in the form of parameterized topologies [13]
			Six human-competitive results duplicating the functionality of 21st-century patented inventions [13]
Long (29-day) runs of 1,000-node Pentium II parallel machine	2002	9.3	Generation of two patentable new inventions [66]

Table 2 lists the five computer systems used to produce the human-competitive results shown in Table 1 that were produced by our research group in the period between 1987 and 2002. Column 6 shows the number of human-competitive results (out of 28 during this period) generated by each computer system.

Table 2 makes the following points:

- There is approximately an order-of-magnitude speed-up (column 3) 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 machine) and the slowest and earliest computer system in the table (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.

There are four successive (approximately) order-of-magnitude increases in computer power in Table 2.

An additional order-of-magnitude increase was achieved by making extraordinarily long runs on the largest machine (the 1,000-node machine). These runs produced two inventions that were subsequently awarded U.S. patent 6,847,851 [66]. The length of the run that produced the two patentable inventions was 28.8 days—almost an order-of-magnitude increase (9.3 times) over the overall 3.4-day average for typical runs of genetic programming that our group had been making at the time. If this final 9.3-to-1 increase is counted as an additional order-of-magnitude increase in computer power, the overall increase in computer power shown in Table 2 is 130,660-to-1 (i.e., about five orders of magnitude).

Table 3 is organized around the five just-explained order-of-magnitude increases in the expenditure of computing power. Column 4 characterizes the qualitative nature of the results produced by genetic programming. The table shows the progression of qualitatively more substantial results produced by genetic programming in terms of five order-of-magnitude increases in the expenditure of computational resources.

The five order-of-magnitude increases in computer power shown in Table 3 correspond closely (albeit not perfectly) with the following progression of qualitatively more substantial results produced by genetic programming:

- toy problems,
- human-competitive results not related to patented inventions,
- twentieth century patented inventions,
- twenty-first 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.

7 Future prospects for producing human-competitive results using genetic programming

The increased availability of computing power should result in an increasing flow of more human-competitive results, as well as more impressive results, in the future.

The length of the runs involving the six post-2000 patented circuits mentioned in Sect. 6 of this paper was 80 h (3.3 days) or about 10^{17} cycles (i.e., 100 petacycles). The relentless iteration of Moore's law promises increased availability of computational resources in future years. If available computer capacity continues to double approximately every 18 months over the next decade or so, a computation requiring 80 h will require only about 1% as much computer time (i.e., about

48 min) a decade from now. That same computation will require only about 0.01% as much computer time (i.e., about 48 seconds) in two decades. Thus, looking forward, we believe that genetic programming can be expected to be increasingly used to automatically generate ever-more complex human-competitive results.

Since its early beginnings, the field of genetic and evolutionary computation has produced a cornucopia of results. Genetic programming and other methods of genetic and evolutionary computation may be especially productive in areas having some or all of the following characteristics:

- where finding the size and shape (i.e., topology) of the ultimate solution is a major part of the problem,
- where the relevant variables are interrelated in highly non-linear ways,
- where the interrelationships among the relevant variables are unknown or poorly understood (or where it is suspected that the prevailing understanding may be wrong),
- where conventional mathematical analysis does not provide an analytic solution,
- where an approximate solution is acceptable (or is the only result that is ever likely to be obtained),
- where small improvements in performance are highly prized,
- where large amounts of primary data requiring examination, classification, and integration is accumulating in computer readable form,
- where humans have no idea how to program a solution, but where the objective is clear, and
- where there are good simulators to measure the performance of the candidate solutions in the genetic population.

References

1. A.M. Turing, Intelligent machinery. [Reprinted in ed. by D.C. Ince, *Mechanical Intelligence: Collected Works of A. M. Turing*. (North Holland, Amsterdam, 1992), pp. 107–127). Also reprinted in ed. by B. Meltzer, D. Michie. 1969]. *Machine Intelligence 5*. (Edinburgh University Press, Edinburgh, 1948)
2. A.M. Turing, Computing machinery and intelligence. Mind **59**(236), 433–460 (1950) [Reprinted in D. C. Ince (ed), *Mechanical Intelligence: Collected Works of A. M. Turing*. (North Holland, Amsterdam, 1992), pp. 133–160]
3. J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. (University of Michigan Press, Ann Arbor, MI, 1975). [Second edition. The MIT Press, Cambridge, MA, 1992]
4. J.R. Koza, in *Hierarchical Genetic Algorithms Operating on Populations of Computer Programs*. Proceedings of the 11th International Joint Conference on Artificial Intelligence, vol. I. (San Mateo, CA: Morgan Kaufmann, 1989), pp. 768–774
5. J.R. Koza, *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*. Stanford University Computer Science Department technical report STAN–CS–90–1314, 1990

6. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT, Cambridge, 1992)
7. J.R. Koza, J.P. Rice, *Genetic Programming: The Movie* (MIT, Cambridge, 1992)
8. J.R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs* (MIT, Cambridge, 1994)
9. J.R. Koza, *Genetic Programming II Videotape: The Next Generation* (MIT, Cambridge, 1994)
10. J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving* (Morgan Kaufmann, San Francisco, 1999)
11. J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, B. Scott, *Genetic Programming III Videotape: Human-Competitive Machine Intelligence* (Morgan Kaufmann, San Francisco, 1999)
12. J.R. Koza, M.A. Keane, M.J. Streeter, Evolving inventions. Sci. Am. **288**(2), 52–59 (2003). (2003)
13. J.R. Koza, M.A. Keane, M.J. Streeter, W. Mydlowec, J. Yu, G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence* (Kluwer, Norwell, 2003)
14. J.R. Koza, M.A. Keane, M.J. Streeter, W. Mydlowec, J. Yu, G. Lanza, D. Fletcher, *Genetic Programming IV Video: Routine Human-Competitive Machine Intelligence* (Kluwer, Norwell, 2003)
15. W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, *Genetic Programming—An Introduction*. (San Francisco, CA: Morgan Kaufmann and Heidelberg: dpunkt, 1998)
16. J.H. Holland, J.S. Reitman, Cognitive Systems Based on Adaptive Algorithms, in *Pattern-Directed Inference Systems*, ed. by D.A. Waterman, F. Hayes-Roth (Academic Press, New York, 1978), pp. 313–329
17. J.H. Holland, K.J. Holyoak, R.E. Nisbett, P.A. Thagard, *Induction Processes of Inference Learning and Discovery* (MIT, Cambridge, 1986)
18. S.F. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. Ph.D. dissertation, University of Pittsburgh, 1980
19. R. Forsyth, BEAGLE—A Darwinian approach to pattern recognition. Kybernetes **10**, 159–166 (1981)
20. N.L. Cramer, in *A Representation for the Adaptive Generation of Simple Sequential Programs*. Proceedings of an International Conference on Genetic Algorithms and Their Applications (Lawrence Erlbaum, Pittsburgh, l985)
21. J.F. Hicklin, *Application of the Genetic Algorithm to Automatic Program Generation*. M. S. thesis, Computer Science Dept., University of Idaho, 1986
22. C. Fujiki, *An Evaluation of Holland's Genetic Algorithm Applied to a Program Generator*. M. S. thesis, Computer Science Dept., University of Idaho, 1986
23. C. Fujiki, J. Dickinson, in *Using the Genetic Algorithm to Generate LISP Source Code to Solve the Prisoner's Dilemma*. Proceedings of the Second International Conference on Genetic Algorithms (Lawrence Erlbaum, Pittsburgh, l987)
24. H. J. Antonisse, K. Keller, in *Genetic Operators for High-Level Knowledge Representations*. Proceedings of the Second International Conference on Genetic Algorithms (Lawrence Erlbaum, Pittsburgh, 1987)
25. A.S. Bickel, R.W. Bickel, in *Tree Structured Rules in Genetic Algorithms*. Proceedings of the Second International Conference on Genetic Algorithms. (Lawrence Erlbaum, Pittsburgh, l987)
26. K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P.L. Lanzi, L. Spector, T. Lee, T.D. Andrea, A. Tyrrell (eds.), in *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 2004*. Proceedings, Part I. Lecture Notes in Computer Science 3102. (Springer, Berlin, 2004)
27. J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (eds.), in *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28–31, 1996, Stanford University* (MIT, Cambridge, 1996)
28. J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo (eds.), in *Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13–16, 1997, Stanford University*. (Morgan Kaufmann, San Francisco, 1997)
29. J.R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D.B. Fogel, M.H. Garzon, D.E. Goldberg, H. Iba, R. Riolo (eds.), in *Genetic Programming 1998: Proceedings of the Third Annual Conference, July 22–25, 1998, University of Wisconsin, Madison, Wisconsin* (Morgan Kaufmann, San Francisco, 1998)

30. J.J. Grefenstette, *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*. (Lawrence Erlbaum Associates, Hillsdale, 1985)
31. M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, M. Tomassini (eds.), in *Genetic Programming: 8th European Conference, EuroGP 2005, Lausanne, Switzerland, March 30–April 1, 2005, Proceedings*. Lecture Notes in Computer Science 3447. (Springer, Heidelberg, 2005)
32. G. Yu, W. Worzel, R. Riolo (eds.), *Genetic Programming Theory and Practice III* (Springer, New York, 2005)
33. S.-B. Cho, H.X. Nguyen, Y. Shan (eds.), *Proceedings of the First Asian-Pacific Workshop on Genetic Programming* (2003). ISBN 0975172409. www.aspgp.org
34. K. E. Kinnear Jr. (ed.), *Advances in Genetic Programming* (MIT, Cambridge, 1994)
35. P.J. Angeline, K. E. Kinnear Jr. (eds.), *Advances in Genetic Programming 2* (MIT, Cambridge, 1996)
36. L. Spector, W.B. Langdon, U.-M. O'Reilly, P. Angeline (eds.), *Advances in Genetic Programming 3* (MIT, Cambridge, 1999)
37. A.L. Samuel, Some studies in machine learning using the game of checkers. IBM J. Res. Dev. **3**(3), 210–229 (1959)
38. A.L. Samuel, in *AI: Where it has been and where it is going*. Proceedings of the Eighth International Joint Conference on Artificial Intelligence (Morgan Kaufmann, Los Altos, 1983), pp. 1152–1157
39. J.R. Koza, Classifying Protein Segments as Transmembrane Domains Using Architecture-Altering Operations in Genetic Programming, in *Advances in Genetic Programming 2*, ed. by P.J. Angeline, K.E. Kinnear Jr. (MIT, Cambridge, 1996), pp. 155–176
40. J.R. Koza, Evolution of a Computer Program for Classifying Protein Segments as Transmembrane Domains Using Genetic Programming, in *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, ed. by R. Altman, D. Brutlag, P. Karp, R. Lathrop, D. Searls (AAAI Press, Menlo Park, 1994), pp. 244–252
41. J.R. Koza, *Classifying protein segments as transmembrane domains using genetic programming and architecture-altering operations*, in ed. by T. Back, D.B. Fogel, Z. Michalewicz Handbook of Evolutionary Computation. (Institute of Physics Publishing, Bristol; Oxford University Press, New York, 1997). pp. G6.1:1–5
42. J.R. Koza, D. Andre, Automated Discovery of Protein Motifs with Genetic Programming, in *Proceedings of AAAI-95 Fall Symposium Series-Genetic Programming*, ed. by E. Siegel (AAAI Press, Menlo Park, CA, 1995)
43. D. Andre, F.H. Bennett III, J.R. Koza, Discovery by Genetic Programming of a Cellular Automata Rule That is Better than any Known Rule for the Majority Classification Problem, in *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28–31, 1996, Stanford University*, ed. by J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (MIT, Cambridge, 1996), pp. 3–11
44. J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming, in *Artificial Intelligence in Design '96*, ed. by J.S. Gero, F. Sudweeks (Kluwer, Dordrecht, 1996), pp. 151–170
45. J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, Toward Evolution of Electronic Animals Using Genetic Programming, in *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, ed. by C.G. Langton, K. Shimohara (The MIT Press. Pages, Cambridge, MA, 1996), pp. 327–334
46. J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, Automated WYWIWYG Design of Both the Topology and Component Values of Analog Electrical Circuits Using Genetic Programming, in *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28–31, 1996, Stanford University*, ed. by J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (MIT, Cambridge, 1996), pp. 123–131
47. J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, in *Four problems for Which a Computer Program Evolved by Genetic Programming is Competitive with Human Performance*. Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (IEEE Press, 1996), pp. 1–10
48. J.R. Koza, D. Andre, F.H. Bennett III, M.A. Keane, Design of a 96 Decibel Operational Amplifier and Other Problems for Which a Computer Program Evolved by Genetic Programming is Competitive with Human Performance, in *Proceedings of 1996 Japan-China Joint International Workshop on Information Systems*, ed. by M. Gen, W. Zu (Ashikaga Institute of Technology, Ashikaga, Japan, 1996), pp. 30–49

49. J.R. Koza, D. Andre, F.H. Bennett III, M.A. Keane, Evolution of a Low-Distortion, Low-Bias 60 Decibel op amp With Good Frequency Generalization Using Genetic Programming, in *Late Breaking Papers at the Genetic Programming 1996 Conference, Stanford University, July 28–31, 1996*, ed. by J.R. Koza (Stanford University Bookstore, Stanford, 1996), pp. 94–100
50. J.R. Koza, F.H. Bennett III, J.L. Hutchings, S.L. Bade, M.A. Keane, D. Andre, Evolving sorting networks using genetic programming and rapidly reconfigurable field-programmable gate arrays. in *Workshop on Evolvable Systems. International Joint Conference on Artificial Intelligence, Nagoya*, ed. by T. Higuchi, (1997), pp. 27–32
51. J.R. Koza, F.H. Bennett III, J.L. Hutchings, S.L. Bade, M.A. Keane, D. Andre, in *Evolving Sorting Networks Using Genetic Programming and the Rapidly Reconfigurable Xilinx 6216 Field-Programmable Gate Array*. Proceedings of the 31st Asilomar Conference on Signals, Systems, and Computers. (IEEE Press, Piscataway, 1997), pp. 404–410
52. J.R. Koza, F.H. Bennett III, J.L. Hutchings, S.L. Bade, M.A. Keane, D. Andre, in *Evolving Computer Programs Using Rapidly Reconfigurable Field-Programmable Gate Arrays and Genetic Programming*. Proceedings of the ACM Sixth International Symposium on Field Programmable Gate Arrays (ACM Press, New York, 1998), pp. 209–219
53. R. Koza, F.H. Bennett III, J.L. Hutchings, S.L. Bade, M.A. Keane, D. Andre, in *Automatic Programming of a Time-Optimal Robot Controller and an Analog Electrical Circuit to Implement the Robot Controller by Means of Genetic Programming*. Proceedings of 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation. (Computer Society Press, Los Alamitos, 1997) pp. 340–346
54. J.R. Koza, F.H. Bennett III, J. Lohn, F. Dunlap, D. Andre, M.A. Keane, in *Automated Synthesis of Computational Circuits Using Genetic Programming*. Proceedings of the 1997 IEEE Conference on Evolutionary Computation (IEEE Press, Piscataway, 1997), pp. 447–452
55. L. Spector, H. Barnum, H.J. Bernstein, Genetic Programming for Quantum Computers, in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, ed. by J.R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D.B. Fogel, M.H. Garzon, D.E. Goldberg, H. Iba, R. Riolo (Morgan Kaufmann, San Francisco, 1998), pp. 365–373
56. L. Spector, H. Barnum, H.J. Bernstein, Quantum Computing Applications of Genetic Programming, in *Advances in Genetic Programming 3*, ed. by L. Spector, W.B. Langdon, U.-M. O'Reilly, P. Angeline (MIT, Cambridge, 1999), pp. 135–160
57. S. Luke, Genetic Programming Produced Competitive Soccer Softbot Teams for RoboCup97, in *Genetic Programming 1998: Proceedings of the Third Annual Conference, July 22–25, 1998, University of Wisconsin, Madison, Wisconsin*, ed. by J.R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D.B. Fogel, M.H. Garzon, D.E. Goldberg, H. Iba, R. Riolo (Morgan Kaufmann, San Francisco, 1998), pp. 214–222
58. D. Andre, A. Teller, Evolving Team Darwin United, in *RoboCup–98: Robot Soccer World Cup II. Lecture Notes in Computer Science*, vol. 1604, ed. by M. Asada, H. Kitano (Springer, Berlin, 1999), pp. 346–352
59. L. Spector, H. Barnum, H.J. Bernstein, N. Swamy. in *Finding a Better-Than-Classical Quantum AND/OR Algorithm Using Genetic Programming*. IEEE Proceedings of 1999 Congress on Evolutionary Computation. (IEEE Press, Piscataway, 1999), pp. 2239–2246
60. H. Barnum, H.J. Bernstein, L. Spector, Quantum circuits for OR and AND of ORs. J. Phys. A Math. Gen. **33**(45), 8047–8057 (2000)
61. F.H. Bennett III, J.R. Koza, M.A. Keane, J. Yu, W. Mydlowec, O. Stiffelman, Evolution by Means of Genetic Programming of Analog Circuits That Perform Digital Functions, in *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, July 13–17, 1999, Orlando, Florida USA*, ed. by W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (Morgan Kaufmann, San Francisco, 1999), pp. 1477–1483
62. J.R. Koza, F.H. Bennett III, Automatic Synthesis, Placement, and Routing of Electrical Circuits by Means of Genetic Programming, in *Advances in Genetic Programming 3*, vol. chap. 6, ed. by L. Spector, W.B. Langdon, U.-M. O'Reilly, P. Angeline (MIT, Cambridge, 1999), pp. 105–134
63. J.R. Koza, F.H. Bennett III, M.A. Keane, J. Yu, W. Mydlowec, O. Stiffelman, Searching for the Impossible Using Genetic Programming, in *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, July 13–17, 1999, Orlando, Florida USA*, ed. by W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, Vasant Honavar, M. Jakiela, R.E. Smith (Morgan Kaufmann, San Francisco, 1999), pp. 1083–1091

64. J.R. Koza, M.A. Keane, J. Yu, F.H. Bennett III, W. Mydlowec, Automatic creation of human-competitive programs and controllers by means of genetic programming. Genet. Program. Evol. Mach. **1**, 121–164 (2000)
65. M.A. Keane, J.R. Koza, M.J. Streeter, Automatic Synthesis Using Genetic Programming of an Improved General-Purpose Controller for Industrially Representative Plants, in *Proceedings of 2002 NASA/DoD Conference on Evolvable Hardware*, ed. by A. Stoica, J. Lohn, R. Katz, D. Keymeulen, R. Zebulum (IEEE Computer Society, Los Alamitos, 2002), pp. 113–122
66. M.A. Keane, J.R. Koza, M.J. Streeter, *Improved General-Purpose Controllers.* Filed July 12, 2002. U.S. Patent 6,847,851, 2004, Issued January 25, 2005
67. M.J. Streeter, M.A. Keane, J.R. Koza, Routine Duplication of Post-2000 Patented Inventions by Means of Genetic Programming, in *Genetic Programming: 5th European Conference, EuroGP 2002, Kinsale, Ireland, April 2002 Proceedings*, ed. by J.A. Foster, E. Lutton, J. Miller, C. Ryan, A.G.B. Tettamanzi (Springer, Berlin, 2002), pp. 26–36
68. L. Spector, H.J. Bernstein, Communication Capacities of Some Quantum Gates, Discovered in Part Through Genetic Programming, in *Proceedings of the Sixth International Conference on Quantum Communication, Measurement, and Computing*, ed. by J.H. Shapiro, O. Hirota (Rinton Press, Princeton, 2003), pp. 500–503
69. J.D. Lohn, G.S. Hornby, D.S. Linden, An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission, in *Genetic Programming Theory and Practice II*, vol. chap. 18, ed. by U.-M. O'Reilly, R.L. Riolo, G. Yu, W. Worzel (Kluwer, Boston, 2004)
70. J.D. Lohn, G.S. Hornby, D.S. Linden, Human-competitive evolved antennas. Artif. Intell. Eng. Des. Anal. Manuf. **22**, 235–247 (2008)
71. L. Spector, *Automatic Quantum Computer Programming: A Genetic Programming Approach* (Kluwer, Boston, 2004)
72. A. Fukunaga, Evolving Local Search Heuristics for SAT Using Genetic Programming, in *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 2004. Proceedings, Part I. Lecture Notes in Computer Science 3102*, ed. by K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P.L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell (Springer, Berlin, 2004)
73. A. Fukunaga, in *Automated Discovery of Composite SAT Variable-Selection Heuristics*. Proceedings of American Association for Artificial Intelligence Conference (2002)
74. H. Lipson, How to Draw a Straight Line Using a GP: Benchmarking Evolutionary Design Against 19th Century Kinematic Synthesis, in *Genetic and Evolutionary Conference 2004 Late-Breaking Papers*, ed. by M. Keijzer (International Society for Genetic and Evolutionary Computation, CD ROM, Seattle, 2004)
75. H. Lipson, Evolutionary synthesis of kinematic mechanisms. Artif. Intell. Eng. Des. Anal. Manuf. **22**, 195–205 (2008)
76. B. Khosraviani, E.R. Levitt, J.R. Koza, Organization Design Optimization Using Genetic Programming, in *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 2004. Proceedings, Part I. Lecture Notes in Computer Science 3102*, ed. by K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P.L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, A. Tyrrell (Springer, Berlin, 2004)
77. B. Lam, V. Ciesielski, *Discovery of Human-Competitive Image Texture Feature Programs Using Genetic programming* (2004)
78. L. Sekanina, R. Ruzicka, *Evolvable Components: From Theory to Hardware Implementations* (Springer, Berlin, 2004)
79. S. Preble, H. Lipson, M. Lipson, Two-dimensional photonic crystals designed by evolutionary algorithms. Appl. Phys. Lett. **86**, 061111 (2005)
80. J.R. Koza, S.H. Al-Sakran, L.W. Jones, Automated Re-Invention of Six Patented Optical Lens Systems Using Genetic Programming, in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO–2005*, ed. by H.-G. Beyer, U.-M. O'Reilly, D.V. Arnold, W. Banzhaf, C. Blum, E.W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J.A. Foster, E.D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G.R. Raidl, T. Soule, A. Tyrrell, J.-P. Watson, E. Zitzler (ACM Press, New York, 2005), pp. 1953–1960

81. J.R. Koza, S.H. Al-Sakran, L.W. Jones, Automated ab initio synthesis of complete designs of four patented optical lens systems by means of genetic programming. Artif. Intell. Eng. Des. Anal. Manuf. **22**, 249–273 (2008)
82. P. Massey, J.A. Clark, S. Stepney, Evolution of a Human-Competitive Quantum Fourier Transform Algorithm Using Genetic Programming, in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO–2005*, ed. by H.-G. Beyer, U.-M. O'Reilly, D.V. Arnold, W. Banzhaf, C. Blum, E.W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J.A. Foster, E.D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G.R. Raidl, T. Soule, A. Tyrrell, J.-P. Watson, E. Zitzler (ACM Press, New York, NY, 2005)
83. F. Corno, E. Sanchez, G. Squillero, Evolving assembly programs: how games help microprocessor validation. IEEE Trans. Evol. Comput. (Special Issue on Evolutionary Computation and Games) (2005)
84. M. Sipper, Y. Azaria, A. Hauptman, Y. Shichel, Attaining human-competitive game playing with genetic programming. IEEE Trans. Syst. Man Cybern. (2005)
85. Y. Azaria, M. Sipper, GP-Gammon: Genetically programming backgammon players. Genet. Program. Evol. Mach. (2005). http://www.cs.bgu.ac.il/~sipper/papabs/gpgammon.pdf
86. Y. Azaria, M. Sipper, GP-Gammon: Using Genetic Programming to Evolve Backgammon Players, in *Proceedings of 8th European Conference on Genetic Programming (EuroGP2005). Volume 3447 of Lecture Notes in Computer Science*, ed. by M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, M. Tomassini (Springer, Heidelberg, 2005), pp. 132–141
87. A. Hauptman, M. Sipper, GP-EndChess: Using Genetic Programming to Evolve Chess Endgame Players, in *Proceedings of 8th European Conference on Genetic Programming (EuroGP2005), vol. 3447 of Lecture Notes in Computer Science*, ed. by M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, M. Tomassini (Springer, Heidelberg, 2005), pp. 120–131
88. Y. Shichel, E. Ziserman, M. Sipper, GP-Robocode: Using Genetic Programming to Evolve Robocode Players, in *Proceedings of 8th European Conference on Genetic Programming (EuroGP2005). Volume 3447 of Lecture Notes in Computer Science*, ed. by M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, M. Tomassini (Springer, Heidelberg, 2005), pp. 143–154
89. N.B. Ho, J. C. Tay. *IEEE Congress on Evolutionary Computation (CEC 2005)*. Edinburgh, Scotland
90. D. Howard, K. Kolibal, *Solution of Differential Equations with Genetic Programming and the Stochastic Bernstein Interpolation*. Biocoumputing-Developmental Systems Group, University of Limerick Technical Report No. BDS-TR-2005-001, Ireland, 2005
91. H.G. Beyer, U.-M. O'Reilly, D.V. Arnold, W. Banzhaf, C. Blum, E.W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J.A. Foster, F.D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G.R. Raidl, T. Soule, A. Tyrrell, J.-P. Watson, E. Zitzler (eds), in *Determining Equations for Vegetation Induced Resistance using Genetic Programming*. Proceedings of the Genetic and Evolutionary Computation Conference GECCO–2005. (ACM Press, New York, 2005)
92. M.J. Baptist, *Modeling Floodplain Biogeomorphology Resistance using Genetic Programming* (DUP Science, Delft, 2005)
93. J.R. Koza, M.A. Keane, M.J. Streeter, *Human-Competitive Automated Engineering Design and Optimization by Means of Genetic Programming*, in ed. by W. Annicchiarico, J. Periaux, M. Cerrolaza, G. Winter. Evolutionary Algorithms and Intelligent Tools in Engineering Optimization. (Wit Pr/Computational Mechanics, 2005)
94. G. Olague, S. Cagnoni, E. Lutton (eds), Introduction to the Special Issue on Evolutionary Computer Vision and Image Understanding. Pattern Recogn. Lett. Elsevier Science (to appear) (2010)
95. G. Olague, E. Lutton, S. Cagnoni (eds.), Evolutionary computer vision. Evol. Comput. MIT Press, (In preparation) (2010)
96. A. Hauptman, M. Sipper, Evolution of an Efficient Search Algorithm for the Mate-in-N Problem in Chess, in *Proceedings of the 10th European Conference on Genetic Programming. Lecture Notes in Computer Science number 4445*, ed. by M. Ebner, M. O'Neill, A. Ekart, L. Vanneschi, A. Esparcia, I. Anna (Springer, Heidelberg, 2007), pp. 78–89
97. R. Cummins, C. O'Riordan, Evolving local and global weighting schemes in information retrieval. Inform. Retriev. **9**(3), 311–330 (2006)
98. R. Cummins, C. O'Riordan, An Analysis of the Solution Space for Genetically Programmed Term-Weighting Schemes in Information Retrieval, in *17th Artificial Intelligence and Cognitive Science Conference (AICS 2006)*, ed. by P.M.D. Bell, P. Sage (Queen's University, Belfast, 2006)

99. R. Cummins, C. O'Riordan, in *Term-Weighting in Information Retrieval Using Genetic Programming: A Three Stage Process*. The 17th European Conference on Artificial Intelligence, ECAI-2006. Riva del Garda, Italy, August 28th–September 1st 2006
100. A. Raja, R.M.A. Azaad, C. Flanagan, C. Ryan, Real-Time, in *Non-Intrusive Evaluation of VoIP*, ed. by M. Ebne, M. O'Neill, A. Ekart, L. Vanneschi, E. Alcazar, A. Isabel. Proceedings of the 10th European Conference on Genetic Programming. Lecture Notes in Computer Science number 4445. (Springer, Heidelberg, 2007), pp. 217–228
101. J. Bongard, H. Lipson, Automated reverse engineering of nonlinear dynamical systems. Proc. Natl. Acad. Sci. USA **104**(24), 9943–9948 (2007)
102. S.Y. El-Bakry, A. Radi, Genetic Programming approach for electron-alkali-metal atom collisions. Int. J. Modern Phys. B **20**(32), 5463–5471 (2006)
103. A. Radi, Prediction of non-linear system in optics using genetic programming. Int. J. Modern Phys. C (2007)
104. M.Y. El-Bakry, A. Radi, Genetic programming approach for flow of steady state fluid between two eccentric spheres. Manuscript accepted for publication, submission to Appl. Rheol. (2007)
105. L. Spector, D.M. Clark, I. Lindsay, B. Barr, J. Klein, in *Genetic Programming for Finite Algebras*. Genetic and Evolutionary Computation Conference 2008, (2008)
106. L. Spector, J. Klein, Machine invention of quantum computing circuits by means of genetic programming. Artif. Intell. Eng. Des. Anal. Manuf. **22**, 275–283 (2008)
107. R. Stadelhofer, W. Banzhaf, D. Suter, Evolving blackbox quantum algorithms using genetic programming. Artif. Intell. Eng. Des. Anal. Manuf. **22**, 285–297 (2008)
108. H. Jianjun, D.G. Erik, L. Shaobo, R. Rosenberg, Automated synthesis of mechanical vibration absorbers using genetic programming. Artif. Intell. Eng. Des. Anal. Manuf. **22**, 207–217 (2008)
109. W. Weimer, T. Nguyen, C. Le Goues, S. Forrest, in *Automatically Finding Patches Using Genetic Programming*. 31st International Conference on Software Engineering (ICSE) (2009)
110. S. Forrest, W. Weimer, T. Nguyen, C. Le Goes., in *A Genetic Programming Approach to Automated Software Repair*. Genetic and Evolutionary Computation Conference (2009)
111. A. Hauptman, A. Elyasaf, M. Sipper, A. Karmon, in *GP-Rush: Using genetic Programming to Evolve Solvers for the Rush Hour Puzzle*. Genetic and Evolutionary Computation Conference. July 2009. Montreal, Canada (2009)
112. C.B. Perez, G. Olague, *Learning Invariant Region Descriptor Operators with Genetic Programming and the F-Measure*. International Conference on Pattern Recognition. Pages 1–4. December 8–11, 2008, (2009). ISBN: 978-1-4244-2174-9
113. C.B. Perez, G. Olague, in *Evolutionary Learning of Local Descriptor Operators for Object Recognition*. Genetic and Evolutionary Computation Conference, July 8–12, 2009 (2009)
114. P. Balasubramaniam, A. Kumar, V. Antony, Solution of matrix Riccati differential equation for nonlinear singular system using genetic programming. Genet. Program. Evol. Mach. **10**, 71–89 (2009)
115. M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data. Science **324**(5923), 81–85 (2009)
116. M. Schmidt, H. Lipson, *Solving iterated functions using genetic programming*. GECCO-2009 Late Breaking Papers, 2009
117. J.G. Ziegler, N.B. Nichols, Optimum settings for automatic controllers. Trans. ASME **64**, 759–768 (1942)
118. K.J. Åström, T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd edn. (Instrument Society of America, Research Triangle Park, 1995)
119. S.W. Wilson, The genetic Algorithm and Biological Development, in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, ed. by J.J. Grefenstette (Lawrence Erlbaum Associates, Hillsdale, 1987), pp. 247–251
120. H. Kitano, Morphogenesis for Evolvable Systems, in *Toward Evolvable Hardware. Lecture Notes in Computer Science*, vol. 1062, ed. by E. Sanchez, M. Tomassini (Springer, Berlin, 1996), pp. 99–117
121. F. Gruau, *Cellular Encoding of Genetic Neural Networks*. Technical report 92–21. Laboratoire de l'Informatique du Parallélisme. Ecole Normale Supérieure de Lyon. May 1992 (1992)
122. F. Gruau, Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process, in *Proceedings of the Workshop on Combinations of Genetic Algorithms and Neural Networks 1992*, ed. by J.D. Schaffer, D. Whitley (The IEEE Computer Society Press, Los Alamitos, 1992)

123. F. Gruau, Genetic Synthesis of Modular Neural Networks, in *Proceedings of the Fifth International Conference on Genetic Algorithms*, ed. by S. Forrest (Morgan Kaufmann Publishers Inc., San Mateo, 1993), pp. 318–325
124. F. Gruau, *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD Thesis. Ecole Normale Supérieure de Lyon (1994)
125. F. Gruau, Genetic Micro Programming of Neural Networks, in *Advances in Genetic Programming*, ed. by K.E. Kinnear Jr. (MIT, Cambridge, 1994), pp. 495–518
126. F. Gruau, D. Whitley, Adding learning to the cellular development process: A comparative study. Evol. Comput. **1**(3), 213–233 (1993)
127. D. Whitley, F. Gruau, L. Preatt, Cellular Encoding Applied to Neurocontrol, in *Proceedings of the Sixth International Conference on Genetic Algorithms*, ed. by L.J. Eshelman (Morgan Kaufmann, San Francisco, 1995), pp. 460–467
128. J.R. Koza, in *Discovery of Rewrite Rules in Lindenmayer Systems and State Transition Rules in Cellular Automata Via Genetic Programming*. Symposium on Pattern Formation (SPF–93), Claremont, California. February 13, 1993 (1993)
129. F. Dellaert, R.D. Beer, Toward an Evolvable Model of Development for Autonomous Agent Synthesis, in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, ed. by R. Brooks, P. Maes (The MIT Press, Cambridge, 1994), pp. 246–257
130. H. Hemmi, J. Mizoguchi, K. Shimohara, Development and Evolution of Hardware Behaviors, in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, ed. by R. Brooks, P. Maes (MIT, Cambridge, 1994), pp. 371–376
131. K. Sims, Evolving 3D Morphology and Behavior by Competition, in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, ed. by R. Brooks, P. Maes (MIT, Cambridge, 1994), pp. 28–39
132. J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, Toward Evolution of Electronic Animals Using Genetic Programming, in *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, ed. by C.G. Langton, K. Shimohara (MIT, Cambridge, 1996), pp. 327–334
133. J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, Reuse, Parameterized Reuse, and Hierarchical Reuse of Substructures in Evolving Electrical Circuits Using Genetic Programming, in *Proceedings of International Conference on Evolvable Systems: From Biology to Hardware (ICES–96). Lecture Notes in Computer Science*, vol. 1259, ed. by T. Higuchi, M. Iwata, W. Liu (Springer, Berlin, 1996), pp. 312–326
134. Scott Brave, Evolving Deterministic Finite Automata Using Cellular Encoding, in *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28–31, 1996, Stanford University*, ed. by J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (MIT, Cambridge, 1996), pp. 39–44
135. E. Tunstel, M. Jamshidi, On genetic programming of fuzzy rule-based systems for intelligent control. Int. J. Intell. Autom. Soft Comput. **2**(3), 273–284 (1996)
136. L. Spector, K. Stoffel, Ontogenetic Programming, in *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28–31, 1996, Stanford University*, ed. by J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (MIT, Cambridge, 1996), pp. 394–399
137. L. Spector, K. Stoffel, Automatic Generation of Adaptive Programs, in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, ed. by P. Maes, M.J. Mataric, J.-A. Meyer, J. Pollack, S.W. Wilson (MIT, Cambridge, 1996), pp. 476–483
138. S. Luke, L. Spector, Evolving Graphs and Networks With Edge Encoding: Preliminary Report, in *Late-Breaking Papers at the Genetic Programming 1996 Conference*, ed. by J.R. Koza (Stanford University Bookstore, Stanford, 1996), pp. 117–124
139. W. Comisky, J. Yu, J. Koza, in *Automatic Synthesis of a Wire Antenna Using Genetic Programming*. Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference, Las Vegas, Nevada (2000), pp. 179–186