



Provided by the author(s) and University College Dublin Library in accordance with publisher policies. Please cite the published version when available.

Title	Recent patents on genetic programming
Author(s)	O'Neill, Michael; Brabazon, Anthony
Publication Date	2009-01
Publication information	Recent Patents on Computer Science, 2 (1): 43-49
Publisher	Bentham Science Publishers
Link to publisher's version	http://www.bentham.org/cseng/samples/cseng2-1/0005CSENG.pdf ; http://www.bentham.org/cseng/index.htm
This item's record/more information	http://hdl.handle.net/10197/2570

Downloaded 2012-02-23T17:11:38Z

Some rights reserved. For more information, please see the item record link above.



Recent Patents on Genetic Programming

Michael O'Neill* & Anthony Brabazon

Natural Computing Research & Applications Group
Complex and Adaptive Systems Laboratory
University College Dublin
Belfield, Dublin 4
Ireland
m.oneill@ucd.ie, anthony.brabazon@ucd.ie

Abstract

Genetic Programming is a form of Natural Computing which adopts principles from neo-Darwinian evolution to automatically solve problems. It is a model induction method in that both the structure and parameters of the solution are explored simultaneously. Genetic Programming is a particularly interesting method as it is claimed to be an invention machine, producing solutions to problems that are competitive and in some cases superior to those produced by human experts. Its best solutions have become patentable inventions in their own right. In this article we overview some of the recent patents relating to Genetic Programming over the past three years. In light of the number and diversity of patent applications during this period it is clear that Genetic Programming is a vibrant field of research, which is having a significant impact on real-world applications, and is demonstrating clear commercial potential.

Keywords: Genetic Programming, Patents, Evolutionary Computation, Natural Computing

1 Introduction

Genetic Programming (GP) is a machine learning method that can be used to automatically solve problems producing solutions with a complex structure, including executable code. This ability allows GP to be applied to a broad range of problems for example ranging from automatic program induction, trading rule design, creation of art (including music and graphics) to animations and circuit design. GP is one of the most successful forms of machine learning to date as it is now capable of routinely producing solutions to problems that are competitive and in some cases superior to those created by human experts [1, 2]. It has also been claimed that GP is itself an invention machine [2, 3, 4], a claim which is partly founded upon the fact that the outputs of GP have been patentable in their own right [5]. In this article we provide some background on patents in Section 2 followed by a brief introduction to Genetic Programming (Section 3). We then provide an overview of some recent patents relating to Genetic Programming in Section 4 before summarizing our findings (Section 5).

2 Background on Patents

Patents can be defined as ‘the grant of a property right’ which creates a ‘right to exclude others from making, using, offering for sale, or selling or importing the invention’ in the territory covered by the patent legislation [6]. The right to exclude covers both the case of deliberate imitation and the subsequent independent discovery of the patented object by others [7]. Patents also provide a legal mechanism which allows the decoupling of invention from the subsequent manufacture of a product, permitting risk-sharing. One party can bear the risk of invention, while another by

obtaining a license to manufacture the patented product can bear the risk of commercialization. Heald [8] in a development of this point suggests that patents can be considered as lowering the transactions cost of sharing intellectual property, compared to the alternative system of trade secrecy.

The basic rationale underlying patent systems is that decentralization of R&D decisions to individual inventors and organizations leads to a more economically efficient outcome, as inventors have private information on the expected costs and benefits of R&D [9, 10]. From a policy-maker's perspective two conflicting goals underlie the patent system [11], a desire to promote the creation and diffusion of inventions. Creation is encouraged by offering property right incentives to successful inventive efforts and diffusion is encouraged by requiring public disclosure of the nature of the invention in public patent documentation. The creation of property rights has the affect of raising barriers of entry to a portion of product space covered by the patent grant. The creation of this barrier results in two potential social costs arising from a patent system, the creation of market power, and by encouraging a patent race - the wasteful duplication of R&D costs [12]. Offsetting this, requiring disclosure of a patented invention confers a positive externality on a firm's competitors possibly reducing R&D duplication [13].

3 Evolutionary Computation and Genetic Programming

Evolutionary Computation (EC) is based upon neo-Darwinian principles of evolution. It is a population-based approach to search where multiple candidate solutions are maintained in parallel. Genetic search operators are applied to breed high-quality solutions as subsequent generations are created using fitness-based selection. The fitness of a candidate solution is a measure of its quality at solving the problem at hand. Evolutionary algorithms are diverse in the types of problems they can solve. In particular, the Genetic Programming representation is especially flexible in that it can be used to evolve models, algorithms (even alternative Evolutionary algorithms in a meta-heuristic fashion), and all manner of structures ranging from binary strings, digital and analogue circuits to chemical structures and art (including paintings and music).

Since Charles Darwin popularized the theory of Natural Selection, the driving force behind evolution, molecular biology has unraveled some of the mysteries of the components that underpin these earlier theories (e.g., most notably the existence and structure of DNA). Neo-Darwinism, which represents the accumulation of knowledge about the process of evolution at a molecular level in conjunction with Darwin's evolutionary theory, has given rise to a powerful family of problem solving algorithms known collectively as Evolutionary Computation (EC).

The origin of EC can be traced back at least to the beginning of Computer Science [1] with the writings of Turing [14] where he discusses the possibility of "genetical or evolutionary search" as one of the methods that might underpin the "the intelligence of machinery". Implementations of EC existed in the 1950's with the Pioneers of EC popularizing their ideas in subsequent decades including Holland [15], Fogel, Owens and Walsh [16], Rechenberg [17], Schwefel [18], Goldberg [19] and Koza [20]. Fogel [21] presents an interesting collection of some of the pioneering papers in the field.

The natural process of evolution sees species being positively or negatively selected depending on their relative success in surviving and reproducing in their current environment. Differential survival and variety generation during reproduction provide the engine for evolution [22, 23]. EC implements this process in a crude fashion as outlined in Fig. 1. A population of candidate solutions are initialized to some starting state (often this is achieved randomly) and the fitness/quality of each solution is determined. Parents are then selected from this population in a biased manner, such that the better quality individuals are more likely to become parents. A series

of genetic operators can be used in the creation of the children, and this is followed by some replacement strategy which determines which children (if any) are allowed to enter into the next generation. In EC, the equivalent of DNA is the representation or encoding that is adopted by each member of the population (often referred to as an individual or candidate solution) to solve a specific problem type. This individual representation can range from simple fixed-length binary or real-valued strings to more complex data structures such as graphs and even variable-length computer code that can be executed. For example, a simple binary string could be used to encode decisions as to whether or not to include an input variable in a solution. The selected inputs might then be passed into a multi-layer perceptron ANN (MLP) to produce a signal such as to buy or sell a position in a market.

3.1 Genetic Programming

A particularly powerful form of EC, in terms of the flexibility of its representation, and its prowess as a model induction method is Genetic Programming. As a specific example of EC in action we present a brief overview of the mechanics of GP in the following section. There are many useful resources which can be used to discover more about Genetic Programming for the interested reader, including [20, 24, 1, 2, 25, 26, 27].

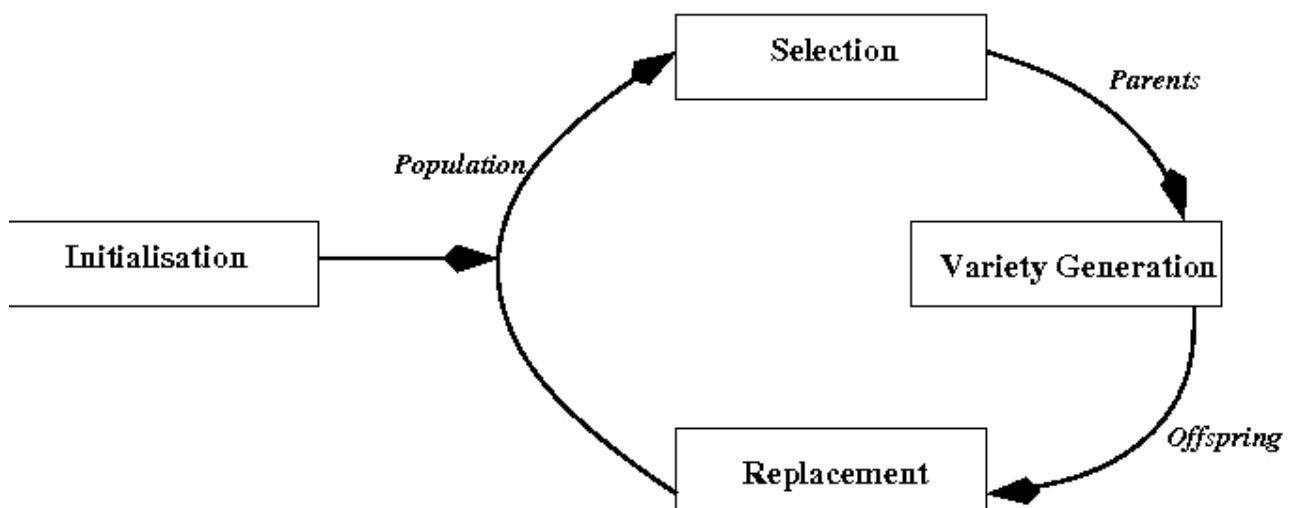


Figure1: The process cycle of Evolutionary Computation.

Genetic Programming (GP) traditionally distinguishes itself from other EAs in two fundamental ways. Instead of evolving fixed-length (e.g., binary) strings a variable-length representation is adopted by GP. As GP is a model induction method it is not always known a-priori what the structure or size of a desired solution might be, and to this end the number of components that makeup a candidate solution must itself be open to the process of evolutionary search. Secondly, unlike other EAs which represent an indirect encoding of a potential solution, evolutionary search can be directly applied to the solution (e.g. computer code in the form of a Lisp S-expression).

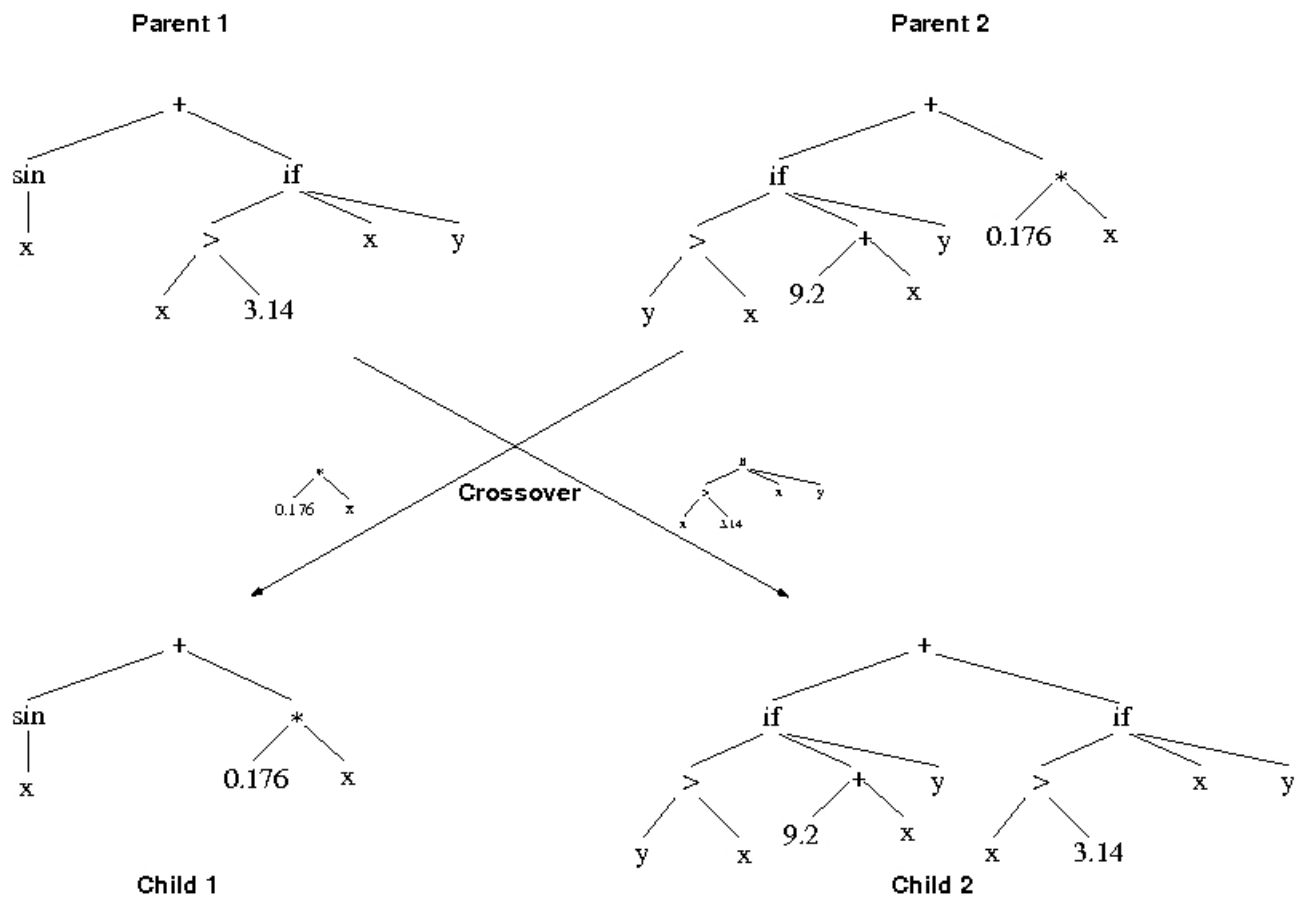


Figure2: Individuals of a Genetic Programming Population are typically represented as Syntax Trees of Lisp S-expressions. The figure illustrate show sub-trees of each individual might be exchanged during the process of crossover.

An example of a GP individual in the form of a Lisp S-expression is given in Fig. 2. Here we can also observe how individuals of this form might be manipulated using a form of the genetic search operator of sub-tree crossover. In the example, two sub-trees are selected randomly from a copy of each parent, and those sub-trees are then exchanged between the two parent copies to create two children. Sub-tree mutation operates by randomly deleting a sub-tree and then randomly generating a brand new sub-tree of a random size to replace the original. Using this representation it is possible to incorporate various programming constructs such as conditions, iterations, recursion and modules/functions. It is also common to allow the inclusion and context of use of these constructs to be evolved over time using what are referred to as Architecture Altering Operators [1]. Standard tree-based GP itself has been subject of patents [28, 29, 30, 31, 32, 33, 34, 35, 36, 5, 37, 38], also some linear forms of GP have been patented [39, 40, 41].

4 Recent Patents with Genetic Programming

The past few years have seen an increase in the number of patent applications that adopt Genetic Programming, encompassing areas such as claims of new classification, prediction and design processes to a proposal for its use in the patent application process itself. The following review covers a sample of these applications since 2006 largely drawing upon the search engines provided by the US Patent and Trademarks Office, European Patent Office and Google Patent Search [42, 43, 44].

4.1 GP for Classification

A process to generate classifiers is described by Crnojevic et al. [45] (Method of developing a classifier using adaboost-over-genetic programming) which combines GP with adaptive boosting (AdaBoost) an ensemble learning technique. In ensemble learning a set of classifiers are trained and their outputs combined. AdaBoost modifies the training set that the classifiers are exposed to by increasing the probability of presenting examples that are incorrectly classified. In this case GP is embedded within an AdaBoost loop, where a GP population of N individuals is combined to generate the classifier. Within the main loop a GP population is evolved and the classifier with the lowest error is selected. This current best individual is combined with earlier GP classifiers to produce a Strong Classifier. If the performance of the Strong Classifier is less than 100% (or some other threshold) the loop is repeated and another GP population is evolved. During each iteration of the loop the training example weights are updated to ensure that the examples classified incorrectly have a greater influence on the error measure. The applicants stress the use of small GP tree depth (3-5) and typically recommend a Population Size and number of Generations both of 300. It is claimed that this process generates classifiers that exhibit a high classification accuracy and low computational complexity. Consequently it is claimed that the resulting classifiers are cheap to implement in hardware and often intuitive to understand.

In a subsequent patent Kisacanin and Yoder [46] describe how the method of developing a classifier using adaboost-over-genetic programming is used to process real-time video image data in order to determine the open versus closed eye state of a human. They specifically mention how this process can be embedded in a microprocessor and used as part of a driver monitoring system. The classifiers are trained as before in an off-line process, and the outputs of this process are used to generate individual classification scores which are combined and compared to a threshold to determine eye state.

Hughes [47] claims a method for intelligent monitoring of network threats in which a network threat response engine is described. GP is used to classify whether or not a particular access to the network is a threat.

4.2 GP for Signal Processing

Xiao et al. [48] describe how a Gene Expression Programming (GEP) form of GP can be combined with numerical optimization (e.g. a genetic algorithm or differential evolution) to evolve digital signal processing circuits. A process is described where a main GP loop exists to generate valid circuits. The parameters of each valid circuit is then optimized for performance with an inner loop of another EA such as a Genetic Algorithm or Differential Evolution. Each circuit is then presented to a second fitness calculation to determine its parsimony. The next generation is then created in the standard GP manner. Two variants of GEP (depth and breadth first encodings) are described as possible methods to adopt as the main GP loop. Both of these forms of GEP have been subject to earlier patents in each case [49, 50], and an additional GEP patent exists [41]. A related patent [51] extends Xiao et al.'s process to evolve digital signal processing circuits that can include a configurable infinite logic aggregator. Infinite valued logic is a generalization of boolean logic where the values can vary between 0 and 1 for example.

A method to extract information from a musical signal that adopts GP is presented in [52, 53].

4.3 GP for Prediction

A patent describing a process to predict future production forecasts for petroleum reservoirs is described by Wilkinson, Yu and Castellini [54]. In order to forecast production rates of oil and/or gas from the wells of a petroleum reservoir it is necessary to generate a computer model. These models are necessary due to the unique conditions that are found at each reservoir, which result in production rates varying drastically from reservoir to reservoir [54]. As water, oil and gas are extracted from a reservoir the characteristics of the reservoir can change, and as such a history matching process takes place that updates reservoir descriptor parameters to reflect the evolving conditions. Typically this process is performed manually one variable at a time. The patent describes an approach to automate the history matching process using GP. After the computer model generation a second predictive phase takes place to forecast and manipulate the production that takes place at the reservoir. GP is also used in this phase to automatically generate accurate production forecasts more efficiently than current methods.

Fox [55] describes a process to identify functional biomolecules, and suggests GP as a method to generate models to predict sequence-activity relationships.

4.4 GP for Control

Lefebvre and Kohn [56] describe a method to implement an indirect controller as part of a control system for an industrial plant, where the behaviour of the indirect controller can be modeled using GP. Rosenof et al. [57] continue along these lines with a claim for a method to calculate marginal cost curves using these plant control models for electricity generation. Marginal cost curves show the variable cost (cost of fuel and for emissions credits) versus the electrical output of a plant. Lefebvre and Kohn [58] earlier described a method using GP that can be used to increase the efficiency of fossil fuel boilers, specifically in terms of optimizing the Flue Gas Desulfurization (FGD) controller. The FGD is used to scrub highly toxic sulphur-based chemicals from the emitted gases, which could produce sulphuric acid in human lungs.

GP is described as one method which can be used in control of switching power supplies [59, 60]. These switching converters are used to convert input DC at one voltage to output DC at another voltage. The operation of the proposed switching converters are optimized using GP.

4.5 GP, Neural Networks and Spectral Data

Tomkins [61] describe a GP approach to generate neural networks for application to spectral data. The process described evolves all aspects of the ANN including the number of layers and nodes, the connectivity of the nodes and their bias and weights and also the transfer function.

4.6 GP for Design

There are several notable applications in the area of Design with Genetic Programming, covering such diverse areas as User Interface, Tyre Thread and analog circuit design.

McConaghy [62] describes a process to automatically generate analog circuits using symbolic regression and GP, which may also include a restriction on the search that it only explores canonical functional forms. This restriction is implemented through the use of a grammar.

In [63], Brown describes a process to design tyre threads which have better noise characteristics. In [64], Noubar et al describes a Method and Apparatus for Evolutionary Design which can use GP for an interactive form of Evolutionary Computation that can be used by groups of people or individuals.

A new User Interface (UI) design for the display and selection of video is claimed by Neely et al. [65]. GP is recommended as one of the methods which can be used to determine placement of the items in the UI. Athale and Tirpak [66] claim a method to alter a Graphical User Interface (GUI) to a behaviour model that predicts an expectation of the user. The behaviour model is generated by examining the users interactions with the GUI.

GP is combined with GAs to design masks for the manufacture of semiconductor devices [67, 68]. Masks are used during photolithography to transfer a pattern onto the semiconductor substrate.

Bonebau et al. [69] describe Interactive Evolutionary Computation where the objective function is subjective and that can include the use of any EA including a GP representation. Examples of its use include design of a paper airplane, a supersonic jet, and drug discovery.

Muller [70] claims a method to optimize the surface geometry design of gears using GP. Variables include the form of the tooth flanks, tooth thickness and height, ease-off topography, rolling error, contact-pattern position, and contact path profile.

4.7 GP for Simulation

Peralta and Kalwij [71] claim a new method for developing computationally optimal designs or strategies which adopts GP alongside other methods including Artificial Neural Networks, Tabu Search, and Simulated Annealing. The method is claimed to be useful for reducing the computational time required to solve complex nonlinear optimization problems by creating adaptively evolving surrogate simulators. They also describe another strategy called the Robustness Enhancing Optimizer is claimed to compensate for the uncertain knowledge of reality by combining optimizers including GP with model parameter sensitivity analysis [72].

4.8 GP for Programming

Addison [73] describes a Whole Cell Computer (WCC) architecture which mimics a biological cell. The architecture comprises a network of WCCs which interact with each other. The architecture results in execution which is massively parallel, distributed, multi-threaded and contains no central processing unit. It is claimed to be a form of membrane computing which operates based on stochastic information flow, and that the proposed computational device can be programmed using GP.

A new way to automatically generate computer programs is claimed by Baum [74, 75, 76]. It is claimed that the search space explored by GP is too large to be practical. It is proposed that cumulative progress should be made by “discovering a series of powerful modules that solve intermediate problems, and building a solution to a deep problem on these modules.” Cumulative progress is combined with compact or constrained enough programs to promote generalization. Computational scaffolds and modules. Concepts and subconcepts can be individually trained. Scaffolds appear to be very similar to GP’s Automatically Defined Functions but with some additional constraints. The patent application also describes the use of a Computer Aided Design (CAD) Tool. For example, the CAD tool allows the user to enter objective functions, data, instructions, and to rearrange modules and scaffolds into bigger programs. Baum also claims a method to discover plans such as programs that schedule work in a factory, play games or solve puzzles. The approach uses GP to generate cognitive programs.

Solomon [77] claims a system that automates the self-organization of collections of computational entities. The approach uses GP to automatically program the behaviour of the mobile agents.

Rice [78] outlines a number of perceived deficiencies of GP including that, programs must be syntactically correct, loops, subroutines and conditional blocks cannot be implemented in a customary manner, programs can't be marked to identify sites for application of genetic operators, instructions are not interchangeable, and the environment and instruction set cannot be altered while programs are running. A new approach to GP is claimed which seeks to address these limitations. Features of a programming language, combinations of features and aspects of other programs are objectized and given a unique identifier. The objects are referred to as codons, and codons are executed like serial instructions.

Huelsbergen [79] describes a method and apparatus which can be used to evaluate one or more genetic programs using a just-in-time optimization process.

Libraries of reusable software code have the potential to make software construction more economical. An approach to generate component libraries using GP is presented by Arthur and Polak [80].

Zorn [81] describe the use of GP for the design of a flexible gap coherent electron interferometer junction circuit and for developing algorithms to test these devices which can be used in DNA sequencing.

4.9 GP for Medicine, Bioinformatics and Molecular Biology

A number of machine learning methods including GP are disclosed as being adopted for generating a model of colorectal disease using biomarker levels by Liew et al [82]. Another method to identify cancer-related or lung-related biomarkers is also described in [83, 84, 85].

GP is used as a statistical data analysis tool [86] in a process used to diagnose and grade Gliomas using Proteomics. Gliomas are complex cancers that start in the brain or spine and can involve different cell types and different growth characteristics.

A method to detect biomolecule targets (in this case proteins) with desired properties is presented in [87]. GP is one of many machine learning methods claimed to be capable of being used in the prediction of a biomolecules activity.

Schadt and Lamb [88] use GP to help the discrimination between closely related samples when considering gene expression data. This is part of a process to identify genes and biological pathways associated with common human diseases.

Lanza et al. [89] use GP with a number of other machine learning methods including boosting, support vector machines, and decision trees to model the biological effects of molecules.

Normalization methods for genotyping are described in [90], which include the use of GP. Genotyping identifies a set of genetic markers belonging to an individual. These markers can be indicators of an individual's risk of developing a specific disease. GP can be used to train an ANN during a clustering step of the process.

4.10 GP for Commerce & Finance

Mueller et al. [91] describe an approach to automatically and dynamically generates sell offers for a product using GP based on the prior experience of sell offers and actual sales. It is claimed that the approach learns best practice as to how to make sell offers which yield positive results. In another patent application [92] they describe how the approach can that take order information and make new offers based on this data.

Parson et al. [93] describe how a number of machine learning methods including inductive logic programming and GP can be used in an approach to detect financial fraud.

Recent changes in corporate governance have resulted in management becoming increasingly accountable for the way their companies are run and their underlying IT systems. [94] uses GP along with other machine learning methods such as Artificial Neural Networks as part of an analysis engine to detect IT compliance issues. Issues that may affect compliance may then be detected automatically and then highlighted to management.

4.11 GP for Games

Fish et al. [95] uses GP to generate bingo card decks that appear random but are designed to limit liability. GP is used to design games that have lower churn levels (churn refers to the number of ways a player can win), evenly distribute payouts to jurisdictions and minimize unexpected large payout groups.

4.12 GP for Patents

GP is described as one of a number of Artificial Intelligence methods that could be used in an improved process to facilitate the preparation, submission and examination of a patent application [96]. For example, potential prior art related to the application can be determined automatically and a potential relevance score displayed.

5 Current and Future Developments

As the field of Genetic Programming matures we are witnessing an increase in its use in real world applications. Clear evidence for this is provided in the number and diversity of patent applications that have been filed in recent years and the ever increasing number of application papers at academic conferences in the field, for example, see recent proceedings of the European Conference on Genetic Programming [97, 98] and the other co-located Evo* conferences [99, 100], the IEEE Congress on Evolutionary Computation [101] and the Genetic and Evolutionary Computation Conference [102]. This paper uncovered 6 patent applications to date in 2008, 29 in 2007, and 17 in 2006. Prior to 2006 we outlined 15 patents, many of these relating to patenting the method of Genetic Programming itself with the three earliest patents in 1992 [28, 29, 30].

The generality of the Genetic Programming method is also illustrated by the wide variety of applications captured in this study crossing areas such as its use in the Patent Process itself, to applications in Games, Medicine, Finance, Design, Programming, Signal Processing, Prediction, Simulation and Classification.

While we have captured patents relating to the use of Genetic Programming, it is very difficult to capture patents that are filed as a result of the output of a Genetic Programming process. We expect to see an ever-increasing number of patents applications relating to Genetic Programming, and we expect that many of these will be products of Genetic Programming itself. The diversity and number of applications demonstrates that the field of Genetic Programming is in an exciting and vibrant phase, and we are optimistic for its continued success into the future.

Conflict of Interest

The authors have no conflicts of interest to declare.

Acknowledgements

The authors acknowledge the support of Science Foundation Ireland under grant no. 08/IN.1/I1868.

References

- [1] Koza JR, Andre D, Bennett IIFH, Keane M. Genetic programming 3: Darwinian invention and problem solving. Morgan Kaufmann, 1999.
- [2] Koza JR, Keane M, Streeter MJ, Mydlowec W, Yu J, Lanza G. Genetic programming IV: Routine human-competitive machine intelligence. Kluwer Academic Publishers 2003.
- [3] Koza JR, Keane M, Streeter MJ. Evolving inventions. Scientific American 2003.
- [4] Koza JR, Keane M, Streeter MJ. What's AI done for me lately? Genetic programming's human-competitive results. IEEE Intel Sys 2003; 18(3):25-31.
- [5] Keane, M.A., Koza, J.R., Streeter M.J. (2005).
- [6] United States Patent and Trademark Office. General information concerning patents. <http://www.uspto.gov/web/offices/pac/doc/general/index.html> (accessed 16 September 2008).
- [7] Von Hippel, E. The sources of innovation. Oxford: Oxford University Press. 1988.
- [8] Heald, P. A Transactions Cost Theory of Patent Law. Working Paper, University of Georgia Law School. 2003.
- [9] Scotchmer, S. On the optimality of the patent renewal system. RAND Journal of Economics, 30 (2):181-196. 1999.
- [10] Hopenhayn, H., Mitchell, M. Innovation variety and patent breadth. RAND Journal of Economics, 32(1):251-166. 2001.
- [11] Gallini, N. Patent policy and costly imitation. RAND Journal of Economics, 23(1):52-63.
- [12] Nelson, R., Winter, S. An Evolutionary Theory of Economic Change. Cambridge, Massachusetts: Harvard University Press. 1982.
- [13] Takalo, R., Kannianen, V. Do patents slow down technological progress? Real options in research, patenting, and market introduction. International Journal of Industrial Organization, 18:1105-1127. 2000.
- [14] Turing AM. Intelligent machines. In Ince, D.C. (Ed.) 1992. Mechanical Intelligence: Collected Works of A.M. Turing, pp.107-128. North-Holland. 1948.
- [15] Holland J. Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press, 1975.
- [16] Fogel LJ, Owens AJ, Walsh MJ. artificial intel ligence through simulated evolution. John Wiley, 1966.
- [17] Rechenberg I. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuggart: Frommann-Holzboog, 1973.
- [18] Schwefel H-P. Evolutionstrategie und numerische Optimierung. Dissertation, Technische Universität, Berlin, 1975.
- [19] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley. 1989.
- [20] Koza, J.R. (1992). Genetic Programming. MIT Press.
- [21] Fogel DB. (Ed.) Evolutionary Computation: The Fossil Record. IEEE Press, 1998.
- [22] Darwin C. On the origin of the species by means of natural selection, or the preservation of favoured races in the struggle for life (reprinted 1985), London: Penguin Books, 1859.
- [23] Spencer H. The principles of biology, Volume 1, London and Edinburgh: Williams and Norgate, 1964.
- [24] Koza JR. Genetic Programming II: Automatic discovery of reusable programs. MIT Press, 1994.

- [25] Poli R, William B. Langdon, and Nicholas Freitag McPhee. A field guide to genetic programming. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk>, 2008. (With contributions by J. R. Koza).
- [26] Banzhaf W, Nordin P, Keller, RE, Francone FD. Genetic Programming – An introduction on the automatic evolution of computer programs and its applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1998.
- [27] Langdon WB, Gustafson S, Koza JR. GP Bibliography. <http://www.cs.bham.ac.uk/wbl/biblio/gp-bib-info.html>.
- [28] Koza, J.R.: US5136686 (1992).
- [29] Koza, J.R., Rice, J.P.: US5148513 (1992).
- [30] Koza, J.R., Rice, J.P.: WO9205497 (1992).
- [31] Koza, J.R., Rice, J.P.: WO9325969 (1993).
- [32] Koza, J.R., Rice, J.P.: US5343554 (1994).
- [33] Koza, J.R., Andre, D., Tackett, W.A.: WO9604605 (1996).
- [34] Koza, J.R., Andre, D., Tackett, W.A.: US20006058385 (2000).
- [35] Koza, J.R., Bennett, III F.H., Andre, D., Keane, M.A.: US20036532453 (2003).
- [36] Bennett, F.H., Koza, J.R.: US20026424959 (2002).
- [37] Koza, J.R., Keane, M.A., Yu, J., Bennett, F.H., Mydlowec, W.: US2004030414 (2004).
- [38] Koza, J.R., Andre, D., Tackett, W.A. US5742738 (1998).
- [39] Francone, F.D., Nordin, P., Banzhaf, W.: WO9802825 (1998).
- [40] Francone, F.D., Nordin, P., Banzhaf, W.: US6493686 (2002).
- [41] De Carvalho Ferreira, M.C.: US2002169563 (2002).
- [42] Google Patent Search. <http://www.google.com/patents>
- [43] US Patent and Trademarks Office. <http://patft.uspto.gov/>
- [44] European Patent Office. <http://ep.espacenet.com/>
- [45] Crnojevic, V.S., Schubert, P.: US2008126275 (2008).
- [46] Kisacanin, B., Yoder, E.: US2008126281 (2008).
- [47] Hughes, W.A.: US20070199070 (2007).
- [48] Xiao, W., Hong, Di-An.: WO2008063767 (2008).
- [49] Xiao, W., Tirpak, T.M.: US7127436 (2006).
- [50] Zhou, C., Xiao, W.: WO2006096280 (2006).
- [51] Mohamed, M.A., Xiao, W.: US2008103995 (2008).
- [52] Kobayashi, Y., Takatsuka, S.: US20070095197 (2007).
- [53] Kobayashi, Y.: US20070112558 (2007).
- [54] Wilkinson D.A., Yu, T.: US2008082469 (2008).
- [55] Fox, R.G.: US20070239364 (2007).
- [56] Lefebvre, W.C., Kohn, D.W.: US20070118238 (2007).
- [57] Rosenof, H., Lefebvre, C.W., Kohn, D.W., Spinney, P.: US20060089730 (2006).
- [58] Lefebvre, W.C., Kohn, D.W.: US20060121616 (2006).
- [59] Latham, P.: US20070108953 (2007).
- [60] Latham, P., Canfield, J.C.: US20070114985 (2007).
- [61] Tomkins, B., Nimmo, C.: US2007288410 (2007).
- [62] McConaghy, T.L.: CA2538615 (2007).
- [63] Brown, J.E.(Jr), Song, T.: US2007175565. (2007).
- [64] Noubar, B.A., Malek, K.M, Bufton, N.J., Ficici, S.G., Austin, H.A., Austin, L.J., McClellan, H.E.: US20070282666 (2007).
- [65] Neely, S.R., Kesteloot L., Novotny M., Buchenau, M., Foley, S.A., O’Neil, M.: US20070283276 (2007).

- [66] Athale, A., Tirpak, T.M. US20080010534 (**2008**).
- [67] Tanaka, T., Suga, O., Terasawa, T., Higuchi, R., Sakanashi, H., Nosato, H., Matsunawa, T.: US20070074146 (**2007**).
- [68] Tanaka, T.: US20070074145 (**2007**).
- [69] Bonabeau, E., Anderson, C., Orme, B., Funes, P., Bandte, O., Sullivan, M., Malinchik, S., Rothermich, J.: US20060195204 (**2006**).
- [70] Muller, H., Vogel, O., Dutschk, R., Hunecke, C.: US20060285936 (**2006**).
- [71] Peralta, R.C., Kalwij, I.M.: US20070168328 (**2007**).
- [72] Peralta, R.C., Kalwij, I.M.: US20070129930 (**2007**).
- [73] Addison, E.: US2007094166 (**2007**).
- [74] Baum, E.: US20070245295 (**2007**).
- [75] Baum, E.: US20070016541 (**2007**).
- [76] Baum, E.: US20060184916 (**2006**).
- [77] Solomon, N.E.: US20060167917 (**2006**).
- [78] Rice, T.M., Bennett, G.E.: US20060288345 (**2006**).
- [79] Huelsbergen, L.F.: US20070118832 (**2007**).
- [80] Arthur, W.B., Polak, W.H.: US20070100784 (**2007**).
- [81] Zorn, M.D.: US20070194225 (**2007**).
- [82] Liew, C-C., Han, M., Yager, T., Chao, S., Zheng, R., Zhang, H.: US20070213939 (**2007**).
- [83] Willey, J.C., Crawford, E.L., Mullins, D'A.N.: US20070092893 (**2007**).
- [84] Willey, J.C., Crawford, E.L., Mullins, D'A., N.: US20070092892 (**2007**).
- [85] Willey, J.C., Crawford, E.L., Mullins, D'A., N.: US20070092891 (**2007**).
- [86] Caprioli, R.: US20070031900 (**2007**).
- [87] Gustafsson, C., Govindarajan, S., Emig, R.A., Fox, R.J., Roy, A.K., Minshull, J.S., Davis, C., Cox, A.R., Patten, P.A., Castle, L.A., Siehl, D.L., Gorton, R.L., Chen, T.: US20060205003 (**2006**).
- [88] Schadt, E., Lamb, J.: US20060241869 (**2006**).
- [89] Lanza, G., Duffy, N.P., Boardman, P.: US20060161407 (**2006**).
- [90] Kermani, B.G.: US20060224529 (**2006**).
- [91] Mueller, R.J., Van Luchene, A.S., Heier, J.E., Amorossi, S., Krishna, S., Markowitz, T.: US20060247973 (**2006**).
- [92] Mueller, R.J., Van Luchene, A.S., Heier, J.E., Amorossi, S., Krishna, S., Markowitz, T.: US20060271441 (**2006**).
- [93] Parson, R.D.G., Andersson, J.D.G., Ball, T., Khatib, W., Tschorn, P.R., Muggleton, S.: US20070027674 (**2007**).
- [94] Baldwin, A., Plaquin, D., Murison, N., Beresnevichiene, Y.: US20060155738 (**2006**).
- [95] Fisk, M.G., Bower, D.W., Libby, B.O.: US20060205477 (**2006**).
- [96] Van Luchene, A.S., Mueller, R.J., Alderucci, D.: US20070220041 (**2007**).
- [97] O'Neill, M., Vanneschi, L., Gustafson, S., et al. (Ed.'s). Genetic Programming: Proceedings of EuroGP 2008 the 11th European Conference on Genetic Programming. LNCS 4971. Springer.
- [98] Ebner, M., O'Neill, M., Ekart, A., et al. (Ed.'s). Genetic Programming: Proceedings of EuroGP 2007 the 10th European Conference on Genetic Programming. LNCS 4445. Springer.
- [99] Giacobini, M., Brabazon, A., Cagnoni, S., et al. (Ed's). Applications of Evolutionary Computation: Proceedings of EvoWorkshops 2008. LNCS 4974. Springer.
- [100] Giacobini, M., Brabazon, A., Cagnoni, S., et al. (Ed's). Applications of Evolutionary Computation: Proceedings of EvoWorkshops 2007. LNCS 4448. Springer.
- [101] Wang, J. (Ed.). Proceedings of CEC 2008 the IEEE Congress on Evolutionary Computation 2008. IEEE Press.

[102] Keijzer, M., Antoniol, G., Bates Congdon, C. et al. (Ed.'s). Proceedings of GECCO 2008 the Genetic and Evolutionary Computation Concerence. ACM.