Genetic Programming for Drug Discovery

Technical Report CES-481 ISSN: 1744-8050
26 February 2008

W. B. Langdon
Mathematical and Biological Sciences, University of Essex, Colchester CO4 3SQ, UK

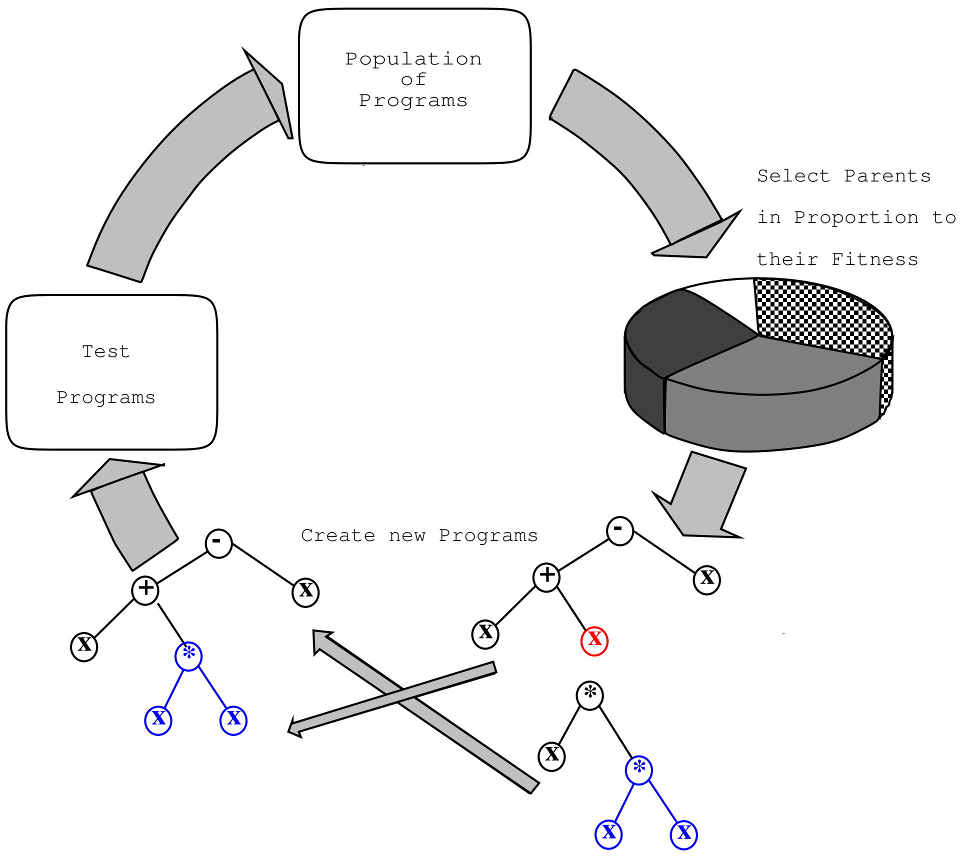

Fig. 1. GP: "Survival of the fittest" program

1. INTRODUCTION

Genetic programming [Koza 1992; Banzhaf et al. 1998; Langdon and Poli 2002] is an established artificial intelligence technique which is used in many areas. Essentially it applies the older genetic algorithm technique [Holland 1992; Goldberg 1989] to automatically generate programs. (See Figure 1.)

The next section will introduce genetic programming, including a very short over view of the many areas where GP has been applied, before Section 3 concentrates on uses of genetic programming in the Pharmaceutical industry. We finish with a worked example (Section 4) which describes an evaluation of machine learning for predicting potential drugs' activity with an pharmaceutically important enzyme (Human cytochrome P450 2D6) made by GlaxoSmithKline chemists [Langdon et al. 2001; Langdon et al. 2002; Langdon et al. 2002; Langdon 2002a; Langdon 2002b; Langdon et al. 2003]. Appendix A gives pointers to further information.

2. WHAT IS GENETIC PROGRAMMING

This section describes genetic programming. Although many variations are possible, GP is essentially simple and you can easily either down load a free version or code your own version and give it a try.

In genetic programming (GP) [Koza 1992; Banzhaf et al. 1998; Langdon 1998; Langdon and Poli 2002] Darwinian natural selection [Darwin 1859] is used inside computers. GP starts with a population containing a primordial soup of randomly created programs. Selection is applied so that the "fitter" programs survive and have more children. Over a series of generations the population evolves and fitness improves. After, say, 50 generations a suitable solution, or approximate solution, may be found. See Figure 1.

2.1 Representing programs in Genetic Programming

Typically in GP the individual programs in the evolving population are stored as parse trees. These are very similar to a Lisp S-expressions. Figure 2 shows a tree which calculates `(a+b)*c`. Where `a, b` and `c` are the program's inputs. The trees internal nodes (`+` and `*`) are functions and the result of the calculation emerges from the tree's root. In data mining and Bioinformatics the programs are usually treated as predictive models. E.g. given properties of a molecule (such as does it have an acidic group, a base or a metallic cation) predict how strongly it will interact with a chosen enzyme. Obviously, as Section 4 will show, real examples are more complex but you can use computers to automatically evolve such programs. The rest of this section describes GP in more detail but in many cases trees like Figure 2 are good enough.

The original GP systems evolved programs represented as trees [Forsyth 1981; Cramer 1985; Bickel and Bickel 1987; Fujiki and Dickinson 1987; Koza 1989] and this is still the most common form of GP. Linear genetic programming is by far the most successful alternatives to trees [Banzhaf et al. 1998]. Typically low level assembler or machine code programs are evolved but grammars allow a linear representation to specify higher level languages, e.g. grammatical evolution [O'Neill and Ryan 2003]. However many other representations have been considered such as Holland's classifiers [De Jong 1987] and graphs, e.g. PADO [Teller and Veloso 1995], Parallel Distributed Genetic Programming [Poli 1996], Cartesian Genetic Programming [Miller and Smith 2006], Binary Decision Diagrams [Downing 2006], Genetic Network Programming [Eguchi et al. 2006] and Multiple Interacting Programs [Angeline 1998].

Efficient ways of using tree programs have been promoted several times, mostly these are based on DAGs and caching, e.g. [Handley 1994a; Ehrenburg 1996; Droste 1997; Kvasnieka and Pospichal 1997; Yangiya 1995; Krawiec and Lijewski 2006]. Another popular approach is to use the evolved program to specify not the final program but how to generate the final solution (be it another program, an architectural design or robot). This goes under several names, e.g. genotype-phenotype mapping, embryology and developmental systems [Gruau 1993; Sims 1994; Keller and Banzhaf 1999; Hemberg and O'Reilly 2004; Ijspeert and Kodjabachian 1999; Hornby et al. 2003; Jacob 2001; Koza et al. 2003; Jones et al. 2005; Kumar 2005; McKay et al. 2006; Miller and Banzhaf 2003; Wilson and Heywood 2006]. Others have rejected evolution of low level languages and sort to evolve programs in as powerful as possible high level languages, either directly [Yu 2001], by using strong typing [Montana 1995] or external grammars to constrain the children of fit programs to have syntactic or semantically desirable properties [Whigham and Crapper 1999; Wong and Leung 1995; Hoai et al. 2003]. The Push system [Spector and Robinson 2002] seeks to avoid human intervention and even (like [Teller 1996]) allows the programs to generate their own breeding operations.

2.2 Genetic Programming Fitness Function

The fitness function guides evolution. It is how you tell evolution which are the better programs. In supervised data mining the classic fitness functions use training examples where the desired output for each example is given along with some (predictive) facts about that example. E.g. given publicly available data and history for a number of stocks traded in the far-east, predict which companies went bankrupt [Zhou 2003]. Or given the solubility of 1000 chemicals and their molecular weight, electric charge and pH, predict the solubility of a new chemical.

Typically the fitness of an individual program in the population is either its root mean squared (RMS) error or the number of correct predictions it made on the training data. However, particularly in Bioinformatics, the available data often contains many more examples of one class (e.g. healthy patients) than of the others. In this case, the easy trap to fall into, is to evolve a predictor which always says the patient is healthy regardless of its inputs. There are a number

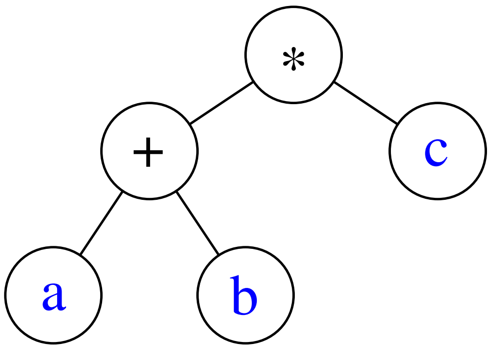

Fig. 2. GP models are often represented as trees. E.g. `eval=(a+b)*c`. Where a,b,c (the inputs to the model) are leaves. While `+` and `*` are functions and internal nodes of the tree. The result emerges from the root node (`*`) of the tree.

of ways of dealing with this, for example using the area under the ROC curve as fitness [Langdon and Buxton 2001a; Langdon and Buxton 2001b] `http://www.cs.ucl.ac.uk/staff/W.Langdon/roc/`.

2.3 Creating new programs

Typically the programs in the initial population are randomly created. There are a number of ways of doing this [Koza 1992; Langdon 2000] however the details may not matter. Any simple mechanism which creates trees of different sizes and shapes and with different functions and inputs, may be fine. Evolution will exert its own biases on the initial population over time [Poli et al. 2007].

It can be interesting to count the number of copies of each input or function there are in the initial population and trace how it evolves over successive generations [Langdon and Poli 1997; Langdon 1998]. This allows you to double check everything you expect to be in the initial population is indeed present. Also, particularly if you have many inputs, it is possible for primitives to become extinct. NB. this may be exactly what is required. For example, in some problems there are many inputs which are not informative and so it may help if they became extinct [Langdon and Buxton 2004]. In data mining we would say that evolution is automatically doing feature selection.

The better programs are selected to be parents for the next generation. The new programs can be mutated copies of their parent or two parents can get together to produce one or more children. Many different ways or randomly mutating programs have been suggested. Generally for data mining it seems to be a good idea to use a few different mutation operators. However try and avoid randomising the child program so much that it has little similarity to its parent.

New children are typically created by copying both parents, extracting a subtree (sub program) from one and inserting into the other (deleting what was their before hand). See Figure 3. The idea is that this sexually produced child will contain parts (genes) from both its highly fit parents and so (occasionally) it may be fitter than either of them. A short animation of Koza's subtree cross over can be found at
`http://www.genetic-programming.com/crossover.gif`

It is common for programs to increase in size over time. This is often known as "bloat" and has been extensively studied [Langdon et al. 1999]. There are many effective ways of dealing with bloat. However simply ensuring that when you create new programs by mutation or crossover they do not exceed a certain size, e.g. 25 inputs, is usually sufficient.

Given the ease with which genetic programming can be used it is no surprise to find it turning up in many applications. Table I gives a brief summary whilst [Langdon and Qureshi 1995; Langdon 1996] contain more information.

3. DRUG DISCOVERY

We start with a brief overview of the drug discovery process and then continue with using GP in drug discovery. Medical research on a disease may discover the disease's life cycle. There may be a critical point in the life cycle that might be disrupted by a drug. E.g. a "target" point where a chemical might bind and prevent the disease's action.

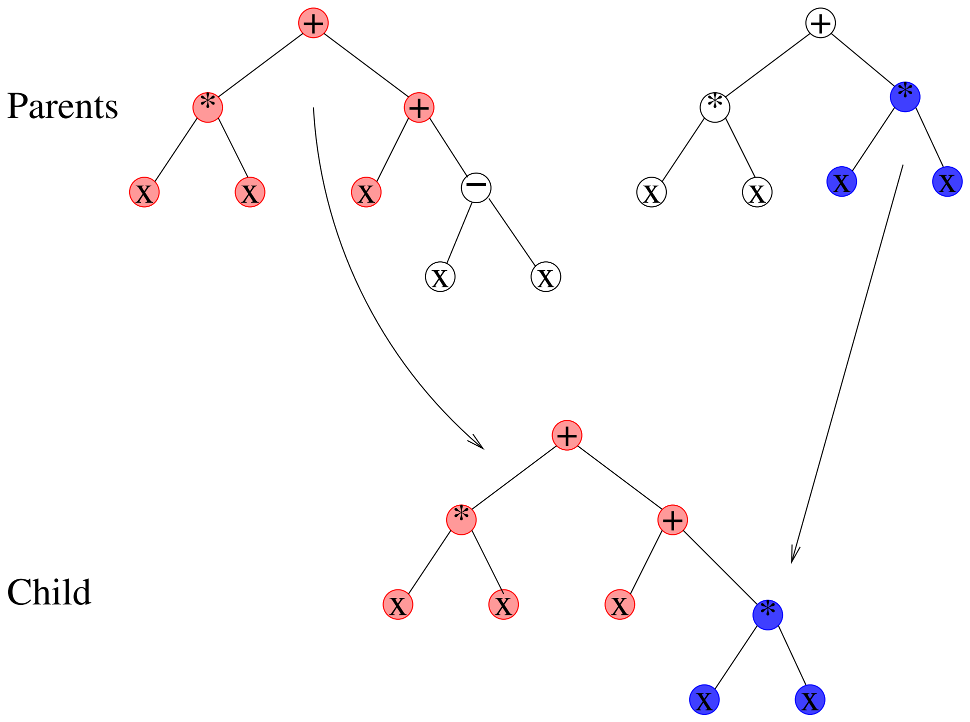

Fig. 3. The new child program is created from fragments of its two parents

The number of potential drugs is huge. It has been estimated there are something like 10^{40} drug like chemicals[1]. Using ultra fast high throughput robot based screening, in the region of a million different chemicals can be measured. E.g. for traces of activity with the target or for potential side effects. It is clear it is impossible to physically measure every potential drug (or even manufacture them all). However computer models can predict properties for some of them. The predictions can guide the choice of which chemicals to look at next and, even if only partially accurate, can save money and time. Computer models of "virtual chemicals", i.e. chemicals that do not exist, are used at many stages of the discovery "funnel" (cf. Figure 4). If the model suggests a virtual chemical looks promising, it can then be made.

3.1 P450 and Drug Discovery

As a drug proceeds down the discovery funnel costs escalate rapidly. NB. the later a potential drug is abandoned the more expensive is the failure. "Late failures" have contributed to the demise or withdraw from the market of several companies and the costs involved have contributed to the trend for pharmaceutical companies to amalgamate. A major cause of late failure is adverse interaction with other parts of the body.

[1]While 10^{40} is a large number, its near the mass of the milky way (cwt), but its nothing to be scared of. In GP the number of halting programs can exceed $10^{100\,000\,000}$ [Langdon and Poli 2006].

Table I. Summary of Uses of Genetic Programming

—Prediction and Classification [MacCallum 2004; Francone et al. 2005].
—Image and Signal Processing [Smith et al. 2005]
  —Armoured vehicle detection [Lin and Bhanu 2005], modelling blood flow [Majeed and Ryan 2006], food spoilage [Ellis et al. 2004].
—Bioinformatics and Pharmaceuticals
  —Using GP within GlaxoSmithKline. In addition to the work in Section 3 onwards, GP has been used for: predicting drug take up into the blood after it has been swallowed [Langdon and Barrett 2004], analysing DNA chip gene expression data [Langdon 2003; Langdon and Buxton 2004]
  —Molecular design [Globus et al. 1999], DNA-based nanotechnology [Feldkamp et al. 2003]
  —Drugs and Structure-activity relationship (SAR) modelling [Archetti et al. 2006] chemistry and protein folding prediction [Handley 1994b; Wasiewicz and Mulawka 2001; Aggarwal and MacCallum 2004]
  —GP and other machine learning techniques used by the pharmaceutical industry are discussed in [Barrett and Langdon 2006].
  —Medicine [Bojarczuk et al. 2000].
—Optimisation
  —Engineering, BAE Systems design [Matthews et al. 2006] General Electric aeroplane turbofan engine optimisation. Engineering [Fan et al. 2002], e.g. design of vehicle suspension [Wang et al. 2005]. Scheduling [Tanev et al. 2002], vehicle routing [Benyahia and Potvin 1998], chemical engineering [Kordon et al. 2005].
—Financial Trading [Brabazon and O'Neill 2004]
  —Currency trading [Neely and Weller 2003], Stock market prediction (Horse race betting) [Tsang et al. 1998; Grosan and Abraham 2006]
—Robots and Autonomous Agents, Artificial Life [Brooks 1992]
  —Economic simulations [Chen 2006], Robot walking [Wolff and Nordin 2003], Robot flying [Augustsson et al. 2002; Spector et al. 2005], Robot hand-eye co-ordination [Langdon and Nordin 2001]
—Artistic
  —GP can suggest novel designs, e.g. to architects [Hemberg and O'Reilly 2004]
  —Evolving L-systems [Jacob 2001; Langdon 2004]
  —Simulations of flocks of birds, bats, herds of wildebeest, etc. These are now state of the art in major Hollywood computer generate film sequences (CGI). Indeed the inventor, Craig Reynolds, was awarded an Oscar[a].

[a] 1997 (70th) Scientific and Engineering Award, Digital Imaging, Craig W. Reynolds for his pioneering contributions to the development of three-dimensional computer animation for motion picture production.

P450 (Figure 5) is a class of enzymes which stick out of animal cells and are attached to the cell wall. They are responsible for degrading most drugs on the market. So it is not too surprising that many companies in the pharmaceutical industry are looking at computer based approaches for early prediction of P450 activity.

4. A WORKED EXAMPLE FROM GLAXOSMITHKLINE

GSK was formed by the merger of Glaxo Wellcome and SmithKline Beecham in 2000. Both GW and SB had large libraries of chemicals which might be used as drugs. Modern robotic chemistry was used to measure the biochemical properties of both libraries, especially the chemicals' interaction with P450. Given the avail-

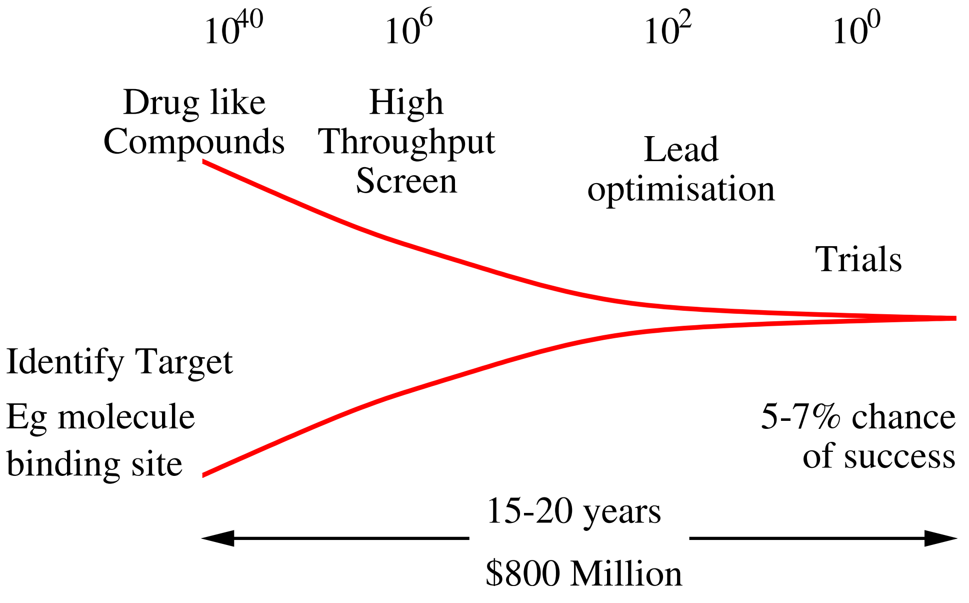

Fig. 4. Drug discovery funnel. The discovery of a drug to treat a disease proceeds in stages. Initially there are many potential drugs. At each stage their number reduces but the investment in each one increases dramatically

ability of this data, GSK wished to see how effective computer models, particularly those based on machine learning, could be at predicting P450 activity. So they held an internal workshop on the feasibility of automatic computer prediction in drug discovery.

Part of the workshop was a blind trial where different groups within GSK were invited to try their techniques. Twelve techniques were entered into the competition. Each was given: former SB training data and former SB data for test. Also they were asked to make predictions, to see how well their technique extrapolated to the P450 activity of chemicals taken from the former GW library.

4.1 P450 data sets

The chemists divided the continuous IC50 activity measurement into three classes: inhibitory, substrate and inactive. Table II summarises the data all the workshop competitors where given, whilst Figure 6 shows the three way split of the IC50 values in the training data.

A common problem in machine learning is learning the supplied training data but then failing to predict unseen data. This is known as “over fitting”. There are essentially two ways to deal with this. Regularisation methods which seek to avoid over fitting by controlling the method’s learning capacity. This limits how much it can learn and so prevents memorising every detail of the training data (which may turn out to be simply noise). Instead it is forced to generalise as best it can the training data. After learning general features of the training data one hopes to be able to apply these lessons to new data. In artificial neural networks (ANN) regularisation is often done by limiting the size (and so learning ability) of

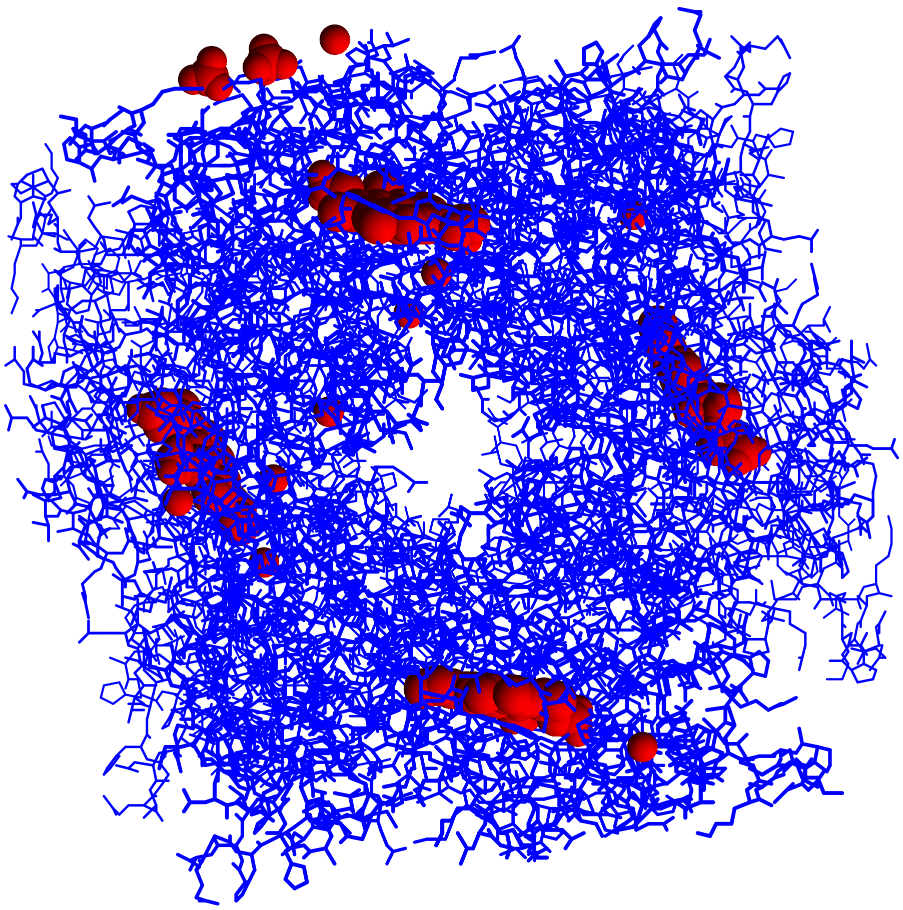

Fig. 5. Human Cytochrome P450 2D6. The main amino acids chain (blue) folds into four clusters leaving a central void. Around the void, in various orientations, are four planar heme groups (larger red clusters). It is believed P450 works by holding drugs and other foreign molecules in the void allowing the heme groups to digest it.

Table II. Summary of P450 Datasets. In addition to 121 features calculated from each chemical's formula the workshop organisers had the measured chemical affinity as concentration. This gives an IC50 activity level, which was used to split the chemicals into 3 classes inhibitor ($\leq 3\mu$M), substrate (> 3 to $< 30\mu$M) and inactive ($\geq 30\mu$M).

Total	Inbitory	Substrate	Inactive	Contents	P450 Activity
3000	1000	1000	1000	part of SB collection for training (random selection but 1000 compounds in each class)	IC50 given
4570	91	1562	2916	part of SB collection for test	with held
1932	114	446	1372	GW compounds to test extrapolation	with held

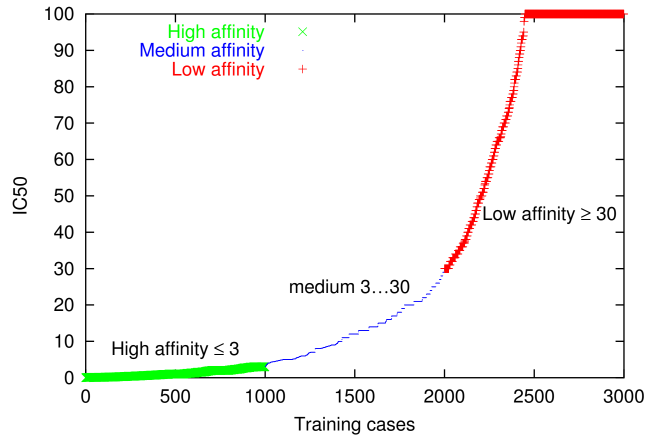

Fig. 6. P450 training data

the network. Over fitting seems to be less of a problem in GP, but one can similarly limit the size of the programs to control the complexity of the evolved models.

It is also common (some would argue essential) to test the prediction of the model on another set of data (which have not been used for training). There are a number of ways of splitting data however they usually assume training and testing data are representative. Part of the idea in the GSK trial was to see how well machine learning could extrapolate from known chemicals to future drugs. It is obviously much more difficult to make predictions about data which will be different from the training data. However, this was why the GW chemicals were not used for training but instead were used to try and see how well the models predicted.

For each chemical GSK calculated 121 "features" from the chemical's formula. The features were either Boolean, categorical or continuous. They represent chemical properties. E.g. charge imbalance, acid or base, heavy metal.

The measurements are high quality but even so the data are still noisy. Note the training data was deliberately set up to have equal numbers in each of the three classes. However the test (or interpolation) data set and the extrapolation set have very unequal splits between the three classes.

4.2 Using Genetic Programming to Predict Drug–P450 Interaction

A simple tree GP system was used [Langdon et al. 2003]. The functions (internal nodes of the tree) were the four arithmetic operators (+, ×, -, and division, protected against divide by zero) and "IF" "min" and "max". The leafs were the 121 features supplied by the chemists plus 100 randomly chosen constant values.

In order to guide the GP we ran each new evolved model on on each of the 3000 chemicals in the training dataset and compared its prediction to the measured P450 activity.[2]

[2]We combined the two common fitness measures, accuracy and RMS error, so that the fitness of the tree was $20\,000 \times \#\text{hits} - \sum \text{error}^2$. Where #hits is the number of chemicals the tree got the class right and error is the difference between the value returned by the tree and measurement.

```
%r14S.p500.2.50 fitness  3.36243e+07 hits 1838 len 51 %Tue Feb 25 17:57:33 2003
(ADD (IFLTE f1
              f2
              (Max
 (ADD
          (DIV
                (SUB f3 20.33)
               0.8546)
         (Min f4 25.58))
 f2)
              (SUB
 (ADD
      (DIV 493.1 f5)
      (SUB f3 23.83))
 20.33))
 (Max
              (Max
                     (Max
                           (Min
                                 (MaxA f4 -21.38)
                                25.58)
                           (IFLTE 0.6186 f6 2 55.27))
                     (Min f7 f8))
              (SUB
                     (MUL 10.02 f2)
                     14.04)))
```

Fig. 7.   Evolved Tree – predicts P450

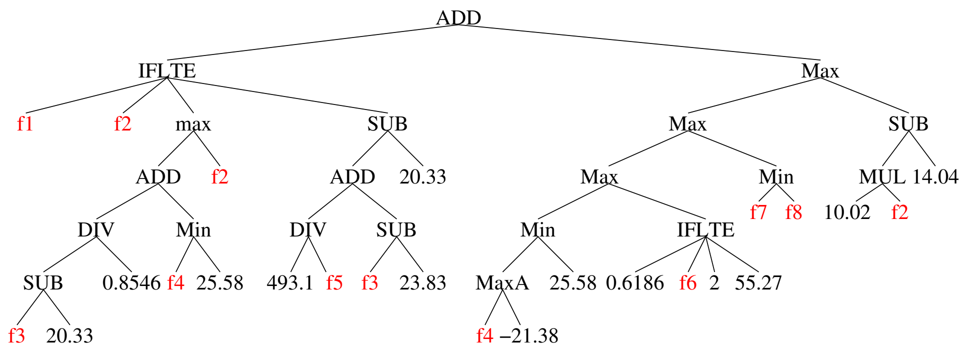

Fig. 8. Evolved Tree – predicts P450. Code in Figure 7 represented as a tree. F1, f2, ... f8 are GSK domain specific features calculated for each chemical from its chemical formula. These 8 features were chosen by GP from the 121 available.

The population contained 5000 trees and the evolution was run for 50 generations. This took about 2 hours. The best model in the final generation is shown in Figure 7, whilst Figure 8 shows it as a tree.

4.3 Results and GSK Workshop Recommendations

Each of the competitors had to use their model to predict the IC50 value and class for each chemical in each data sets. In particular, they had to predict the activity of the 6502 unseen chemicals in the SB test set and GW extrapolation set. Then they had to send their model and its predictions to the competition organisers before the workshop.

The GP model (cf. Figures 7 and 8) produced the most accurate predictions of IC50.

After the workshop, the GSK chemists produced the following recommendations to GSK:

(1) GP, NN and Logistic Regression all showed reasonable predictivity, but far from ideal. They should be explored further.

(2) GSK should follow up on GP methods and make the GP technology available within GSK.

(3) GP produced a more easily understood model, using a small number of features which made sense to the P450 modelling experts.

Before concluding; a short anecdote. We were the last presentation on the second day of the workshop. The chemists had had two days of computer experts talking to them about neural networks, MLPs, ANNs, decision trees, SVM, hidden layers, forests, etc. etc. I also committed the sin of not talking about chemistry and they sat and looked at their watches... until I put up Figure 8. Whereupon the meeting came alive. The model contains chemistry! This provoked a big discussion amongst the chemists about the relative importance of their favourite features.

The lesson – Make your model fit on one slide and show it to the users.

5. CONCLUSIONS

Genetic programming offers no guarantee that it will find a suitable solution within an acceptable amount of time. In practise GP has solved difficult but economically interesting problems (for which it is known that no guarantee is possible). While many of the new techniques require more computation time, computer power is increasingly available. Indeed [Buontempo et al. 2005] and [Deutsch 2003, page 49] shows conventional techniques can be out performed in a few minutes. However, from a commercial perspective, spending computer hours (e.g. over night) rather than man-days is a bargain [Bains et al. 2004].

Genetic programming combines a flexible problem representation with a powerful search mechanism which requires minimal assumptions. It can automatically evolve innovative solutions to many diverse problems, including pharmaceutical problems. It can be used in drug discovery to predict which potential molecules might become drugs long before testing is required. At the GSK workshop GP came top out of 12 entries.

REFERENCES

Aggarwal, V. and MacCallum, R. 2004. Evolved matrix operations for post-processing protein secondary structure predictions. In Genetic Programming 7th European Conference, EuroGP 2004, Proceedings, M. Keijzer et al., Eds. LNCS, vol. 3003. Springer, Coimbra, Portugal, 220–229.

Angeline, P. J. 1998. Multiple interacting programs: A representation for evolving complex behaviors. Cybernetics and Systems 29, 8, 779–806.

Archetti, F., Lanzeni, S., Messina, E., and Vanneschi, L. 2006. Genetic programming for human oral bioavailability of drugs. In GECCO 2006, M. Keijzer et al., Eds. Vol. 1. ACM Press, Seattle, Washington, USA, 255–262.

Augustsson, P., Wolff, K., and Nordin, P. 2002. Creation of A learning, flying robot by means of evolution. In GECCO 2002, W. B. Langdon et al., Eds. New York, 1279–1285.

Bains, W., Basman, A., and White, C. 2004. HERG binding specificity and binding site structure: Evidence from a fragment-based evolutionary computing SAR study. Progress in Biophysics and Molecular Biology 86, 2, 205–233.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. 1998. Genetic Programming – An Introduction. Morgan Kaufmann.

Barrett, S. J. and Langdon, W. B. 2006. Advances in the application of machine learning techniques in drug discovery, design and development. In Applications of Soft Computing: Recent Trends, A. Tiwari et al., Eds. Advances in Soft Computing. Springer, 99–110.

Benyahia, I. and Potvin, J.-Y. 1998. Decision support for vehicle dispatching using genetic programming. IEEE Trans. on Systems, Man, and Cybernetics part A: systems and humans 28, 3, 306–314.

Bickel, A. S. and Bickel, R. W. 1987. Tree structured rules in genetic algorithms. In Genetic Algorithms and their Applications: Proceedings of the second International Conference on Genetic Algorithms, J. J. Grefenstette, Ed. Lawrence Erlbaum Associates, MIT, 77–81.

Bojarczuk, C. C., Lopes, H. S., and Freitas, A. A. 2000. Genetic programming for knowledge discovery in chest-pain diagnosis. IEEE Engineering in Medicine and Biology Magazine 19, 4, 38–44.

Brabazon, A. and O'Neill, M. 2004. Bond-issuer credit rating with grammatical evolution. In Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC, G. R. Raidl et al., Eds. LNCS, vol. 3005. Springer, Coimbra, Portugal, 270–279.

Brooks, R. A. 1992. Artificial life and real robots. In Proceedings of the First European Conference on Artificial Life, F. J. Varela and P. Bourgine, Eds. MIT Press, 3–10.

Buontempo, F. V., Wang, X. Z., Mwense, M., Horan, N., Young, A., and Osborn, D. 2005. Genetic programming for the induction of decision trees to model ecotoxicity data. Journal of Chemical Information and Modeling 45. ASAP article. Web Release Date: May 12, 2005.

Chen, S.-H. 2006. Computationally intelligent agents in economics and finance. Information Sciences. Article in Press.

Cramer, N. L. 1985. A representation for the adaptive generation of simple sequential programs. In Proceedings of an International Conference on Genetic Algorithms and the Applications, J. J. Grefenstette, Ed. Carnegie-Mellon University, Pittsburgh, PA, USA, 183–187.

Darwin, C. 1859. The Origin of Species, Penguin Classics, 1985.

De Jong, K. 1987. On using genetic algorithms to search program spaces. In Genetic Algorithms and their Applications: Proceedings of the second international conference on Genetic Algorithms, J. J. Grefenstette, Ed. Lawrence Erlbaum Associates, MIT, Cambridge, MA, USA, 210–216.

Deutsch, J. M. 2003. Evolutionary algorithms for finding optimal gene sets in microarray prediction. Bioinformatics 19, 1, 45–52.

Downing, R. M. 2006. Evolving binary decision diagrams with emergent variable orderings. In PPSN IX, T. P. Runarsson et al., Eds. LNCS, vol. 4193. Springer, Reykjavik, Iceland, 798–807.

Droste, S. 1997. Efficient genetic programming for finding good generalizing boolean functions. In Genetic Programming 1997: Proceedings of the Second Annual Conference, J. R. Koza et al., Eds. Stanford, 82–87.

EGUCHI, T., HIRASAWA, K., HU, J., AND OTA, N. 2006. A study of evolutionary multiagent models based on symbiosis. *IEEE Trans. on Systems, Man, and Cybernetics, Part B 36*, 1, 179–193.

EHRENBURG, H. 1996. Improved direct acyclic graph handling and the combine operator in genetic programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, J. R. Koza et al., Eds. MIT Press, Stanford University, CA, USA, 285–291.

ELLIS, D. I., BROADHURST, D., AND GOODACRE, R. 2004. Rapid and quantitative detection of the microbial spoilage of beef by fourier transform infrared spectroscopy and machine learning. *Analytica Chimica Acta 514*, 2, 193–201.

FAN, Z., SEO, K., ROSENBERG, R. C., HU, J., AND GOODMAN, E. D. 2002. Exploring multiple design topologies using genetic programming and bond graphs. In *GECCO 2002*, W. B. Langdon et al., Eds. New York, 1073–1080.

FELDKAMP, U., RAUHE, H., AND BANZHAF, W. 2003. Software tools for DNA sequence design. *Genetic Programming and Evolvable Machines 4*, 2, 153–171.

FORSYTH, R. 1981. BEAGLE A Darwinian approach to pattern recognition. *Kybernetes 10*, 159–166.

FRANCONE, F. D., DESCHAINE, L. M., BATTENHOUSE, T., AND WARREN, J. J. 2005. Discrimination of unexploded ordnance from clutter using linear genetic programming. In *Genetic Programming Theory and Practice III*, T. Yu et al., Eds. Springer, Ann Arbor, Chapter 4, 49–64.

FUJIKI, C. AND DICKINSON, J. 1987. Using the genetic algorithm to generate lisp source code to solve the prisoner's dilemma. In *Genetic Algorithms and their Applications: Proceedings of the second international conference on Genetic Algorithms*, J. J. Grefenstette, Ed. Lawrence Erlbaum Associates, MIT, Cambridge, MA, USA, 236–240.

GLOBUS, A., LAWTON, J., AND WIPKE, T. 1999. Automatic molecular design using evolutionary techniques. *Nanotechnology 10*, 3, 290–299.

GOLDBERG, D. E. 1989. *Genetic Algorithms in Search Optimization and Machine Learning*.

GROSAN, C. AND ABRAHAM, A. 2006. Stock market modeling using genetic programming ensembles. In *Genetic Systems Programming: Theory and Experiences*, N. Nedjah et al., Eds. Studies in Computational Intelligence, vol. 13. Springer, Germany, 133–148. Forthcoming.

GRUAU, F. 1993. Cellular encoding as a graph grammar. *IEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives (Digest No.092)*, 17/1–10.

HANDLEY, S. 1994a. On the use of a directed acyclic graph to represent a population of computer programs. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*. Vol. 1. Orlando, Florida, USA, 154–159.

HANDLEY, S. G. 1994b. The prediction of the degree of exposure to solvent of amino acid residues via genetic programming. In *Second International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Stanford University, Stanford, CA, USA.

HEMBERG, M. AND O'REILLY, U.-M. 2004. Extending grammatical evolution to evolve digital surfaces with genr8. In *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, M. Keijzer et al., Eds. LNCS, vol. 3003. Springer, Coimbra, Portugal, 299–308.

HOAI, N. X., MCKAY, R. I., AND ABBASS, H. A. 2003. Tree adjoining grammars, language bias, and genetic programming. In *Genetic Programming, Proceedings of EuroGP'2003*, C. Ryan et al., Eds. LNCS, vol. 2610. Springer, Essex, 335–344.

HOLLAND, J. H. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press. (Uni. Michigan 1975).

HORNBY, G. S., LIPSON, H., AND POLLACK, J. B. 2003. Generative representations for the automated design of modular physical robots. *IEEE trans. on Robotics and Automation 19*, 4, 709–713.

IJSPEERT, A. J. AND KODJABACHIAN, J. 1999. Evolution and development of a central pattern generator for the swimming of a lamprey. *Artificial Life 5*, 3, 247–269.

JACOB, C. 2001. *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann.

JONES, L. W., AL-SAKRAN, S. H., AND KOZA, J. R. 2005. Automated design of a previously patented aspherical optical lens system by means of genetic programming. In *Genetic Programming Theory and Practice III*, T. Yu et al., Eds. Springer, Ann Arbor, Chapter 3, 33–48.

Keller, R. E. and Banzhaf, W. 1999. The evolution of genetic code in genetic programming. In *GECCO-99*, W. Banzhaf et al., Eds. Vol. 2. Orlando, Florida, USA, 1077–1082.

Kordon, A., Castillo, F., Smits, G., and Kotanchek, M. 2005. Application issues of genetic programming in industry. In *Genetic Programming Theory and Practice III*, T. Yu et al., Eds. Springer, Ann Arbor, Chapter 16, 241–258.

Koza, J. R. 1989. Hierarchical genetic algorithms operating on populations of computer programs. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, N. S. Sridharan, Ed. Vol. 1. Morgan Kaufmann, 768–774.

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

Koza, J. R., Keane, M. A., and Streeter, M. J. 2003. What's AI done for me lately? genetic programming's human-competitive results. *IEEE Intelligent Systems 18*, 3, 25–31.

Krawiec, K. and Lijewski, P. 2006. Genetic graph programming for object detection. In *Proceedings 8th International Conference on Artificial Intelligence and Soft Computing ICAISC*, L. Rutkowski et al., Eds. LNAI, vol. 4029. Springer, Zakopane, Poland, 804–813.

Kumar, S. 2005. The art of nature: Evolving mechanisms of development for self- organization and differentiation. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, D. Corne et al., Eds. Vol. 1. Edinburgh, UK, 551–558.

Kvasnieka, V. and Pospichal, J. 1997. Simple implementation of genetic programming by column tables. In *Soft Computing in Engineering Design and Manufacturing*, P. K. Chawdhry et al., Eds. Springer London, 48–56.

Langdon, W. B. 1996. A bibliography for genetic programming. In *Advances in Genetic Programming 2*, P. J. Angeline and K. E. Kinnear, Jr., Eds. MIT Press, Appendix B, 507–532.

Langdon, W. B. 1998. *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!* Kluwer.

Langdon, W. B. 2000. Size fair and homologous tree genetic programming crossovers. *Genetic Programming and Evol. Mach. 1*, 1/2, 95–119.

Langdon, W. B. 2002a. Application of genetic programming in drug lead discovery. 8th Iberoamerican Conference on Artificial Intelligence. Invited conference speaker.

Langdon, W. B. 2002b. A hybrid genetic programming neural network classifier for use in drug discovery. In *Soft Computing Systems - Design, Management and Applications*, A. Abraham et al., Eds. Frontiers in Artificial Intelligence and Applications Vol. 87. IOS Press Amsterdam, Berlin, Oxford, Tokyo, Washington D.C., Universidad de Chile, Chile, 6. Invited conference speaker.

Langdon, W. B. 2003. Predicting cancer. *UCL Science 17*, 2.

Langdon, W. B. 2004. Global distributed evolution of L-systems fractals. In *Genetic Programming, Proceedings of EuroGP'2004*, M. Keijzer et al., Eds. LNCS, vol. 3003. Springer, Coimbra, Portugal, 349–358.

Langdon, W. B. and Barrett, S. J. 2004. Genetic programming in data mining for drug discovery. In *Evolutionary Computing in Data Mining*, A. Ghosh and L. C. Jain, Eds. Studies in Fuzziness and Soft Computing, vol. 163. Springer, Chapter 10, 211–235.

Langdon, W. B., Barrett, S. J., and Buxton, B. F. 2001. Genetic programming for combining neural networks for drug discovery. In *Soft Computing and Industry Recent Applications*, R. Roy et al., Eds. Springer, 597–608. Published 2002.

Langdon, W. B., Barrett, S. J., and Buxton, B. F. 2002. Combining decision trees and neural networks for drug discovery. In *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, J. A. Foster et al., Eds. LNCS, vol. 2278. Springer, Kinsale, Ireland, 60–70.

Langdon, W. B., Barrett, S. J., and Buxton, B. F. 2003. Predicting biochemical interactions – human P450 2D6 enzyme inhibition. In *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, R. Sarker et al., Eds. IEEE Press, Canberra, 807–814.

Langdon, W. B. and Buxton, B. F. 2001a. Evolving receiver operating characteristics for data fusion. In *Genetic Programming, Proceedings of EuroGP'2001*, J. F. Miller et al., Eds. LNCS, vol. 2038. Springer, Lake Como, Italy, 87–96.

Langdon, W. B. and Buxton, B. F. 2001b. Genetic programming for combining classifiers. In GECCO-2001, L. Spector et al., Eds. San Francisco, California, USA, 66–73.

Langdon, W. B. and Buxton, B. F. 2004. Genetic programming for mining DNA chip data from cancer patients. Genetic Programming and Evol. Mach. 5, 3, 251–257.

Langdon, W. B., Buxton, B. F., and Barrett, S. J. 2002. Combining machine learning techniques to predict compounds' cytochrome P450 high throughput screening inhibition. Knowledge Discovery meets Drug Discovery. Poster.

Langdon, W. B. and Nordin, P. 2001. Evolving hand-eye coordination for a humanoid robot with machine code genetic programming. In Genetic Programming, Proceedings of EuroGP'2001, J. F. Miller et al., Eds. LNCS, vol. 2038. Springer, Lake Como, Italy, 313–324.

Langdon, W. B. and Poli, R. 1997. An analysis of the MAX problem in genetic programming. In Genetic Programming 1997: Proceedings of the Second Annual Conference, J. R. Koza et al., Eds. Stanford University, CA, USA, 222–230.

Langdon, W. B. and Poli, R. 2002. Foundations of Genetic Programming. Springer.

Langdon, W. B. and Poli, R. 2006. The halting probability in von Neumann architectures. In Proceedings of the 9th European Conference on Genetic Programming, P. Collet et al., Eds. LNCS, vol. 3905. Springer, Budapest, Hungary, 225–237.

Langdon, W. B. and Qureshi, A. 1995. Genetic programming – computers using "natural selection" to generate programs. Research Note RN/95/76, University College London

Langdon, W. B., Soule, T., Poli, R., and Foster, J. A. 1999. The evolution of size and shape. In Advances in Genetic Programming 3, L. Spector et al., Eds. MIT Press, Chapter 8, 163–190.

Lin, Y. and Bhanu, B. 2005. Object detection via feature synthesis using MDL-based genetic programming. IEEE Trans. on Systems, Man and Cybernetics, Part B 35, 3, 538–547.

MacCallum, R. M. 2004. Striped sheets and protein contact prediction. Bioinformatics 20, Suppl 1 (Aug. 4), I224–I231.

Majeed, H. and Ryan, C. 2006. A re-examination of a real world blood flow modeling problem using context-aware crossover. In Genetic Programming Theory and Practice IV, R. L. Riolo et al., Eds. Springer, Ann Arbor, Chapter 14.

Matthews, P. C., Standingford, D. W. F., Holden, C. M. E., and Wallace, K. M. 2006. Learning inexpensive parametric design models using an augmented genetic programming technique. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 20, 1–18.

McKay, R. I., Hoang, T. H., Essam, D. L., and Nguyen, X. H. 2006. Developmental evaluation in genetic programming: the preliminary results. In Proceedings of the 9th European Conference on Genetic Programming, P. Collet et al., Eds. LNCS, vol. 3905. Springer, Budapest, Hungary, 280–289.

Miller, J. F. and Banzhaf, W. 2003. Evolving the program for a cell: from french flags to boolean circuits. In On Growth, Form and Computers, S. Kumar and P. J. Bentley, Eds. Academic Press.

Miller, J. F. and Smith, S. L. 2006. Redundancy and computational efficiency in cartesian genetic programming. IEEE Trans. on Evolutionary Computation 10, 2, 167–174.

Montana, D. J. 1995. Strongly typed genetic programming. Evolutionary Computation 3, 2, 199–230.

Neely, C. J. and Weller, P. A. 2003. Intraday technical trading in the foreign exchange market. Journal of International Money and Finance 22, 2, 223–237.

O'Neill, M. and Ryan, C. 2003. Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. Kluwer.

Poli, R. 1996. Some steps towards a form of parallel distributed genetic programming. In The 1st Online Workshop on Soft Computing (WSC1). Nagoya University, Japan, http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/.

Poli, R., Langdon, W. B., and Dignum, S. 2007. On the limiting distribution of program sizes in tree-based genetic programming. In Proceedings of the 10th European Conference on Genetic Programming. LNCS. Valencia, Spain. To be presented.

Sims, K. 1994. Evolving 3D morphology and behaviour by competition. In Artificial Life IV Proceedings, R. Brooks and P. Maes, Eds. MIT Press, 28–39.

SMITH, S. L., LEGGETT, S., AND TYRRELL, A. M. 2005. An implicit context representation for evolving image processing filters. In Applications of Evolutionary Computing, EvoWorkshops2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC, F. Rothlauf et al., Eds. LNCS, vol. 3449. Springer, Lausanne, Switzerland, 407–416.

SPECTOR, L., KLEIN, J., PERRY, C., AND FEINSTEIN, M. 2005. Emergence of collective behavior in evolving populations of flying agents. Genetic Programming and Evol. Mach. 6, 1, 111–125.

SPECTOR, L. AND ROBINSON, A. 2002. Genetic programming and autoconstructive evolution with the push programming language. Genetic Programming and Evol. Mach. 3, 1, 7–40.

TANEV, I. T., UOZUMI, T., AND MOROTOME, Y. 2002. An application service provider approach for hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem. In GECCO 2002, W. B. Langdon et al., Eds. New York, 1219–1226.

TELLER, A. 1996. Evolving programmers: The co-evolution of intelligent recombination operators. In Advances in Genetic Programming 2, P. J. Angeline and K. E. Kinnear, Jr., Eds. MIT Press, Chapter 3, 45–68.

TELLER, A. AND VELOSO, M. 1995. Program evolution for data mining. The International Journal of Expert Systems 8, 3, 216–236.

TSANG, E. P. K., LI, J., AND BUTLER, J. M. 1998. EDDIE beats the bookies. Software: Practice and Experience 28, 10, 1033–1043.

WANG, J., FAN, Z., TERPENNY, J. P., AND GOODMAN, E. D. 2005. Knowledge interaction with genetic programming in mechatronic systems design using bond graphs. IEEE Trans. on Systems, Man, and Cybernetics, Part C 35, 2, 172–182.

WASIEWICZ, P. AND MULAWKA, J. J. 2001. Molecular genetic programming. Soft Computing - A Fusion of Foundations, Methodologies and Applications 2, 5, 106–113.

WHIGHAM, P. A. AND CRAPPER, P. F. 1999. Time series modelling using genetic programming: An application to rainfall-runoff models. In Advances in Genetic Programming 3, L. Spector et al., Eds. MIT Press, Chapter 5, 89–104.

WILSON, G. AND HEYWOOD, M. 2006. Probabilistic adaptive mapping developmental genetic programming (PAM DGP): A new developmental approach. In PPSN IX, T. P. Runarsson et al., Eds. LNCS, vol. 4193. Springer, Reykjavik, Iceland, 751–760.

WOLFF, K. AND NORDIN, P. 2003. Learning biped locomotion from first principles on a simulated humanoid robot using linear genetic programming. In GECCO-2003, E. Cantú-Paz et al., Eds. LNCS, vol. 2723. Springer, Chicago, 495–506.

WONG, M. L. AND LEUNG, K. S. 1995. Inducing logic programs with genetic algorithms: the genetic logicprogramming system genetic logic programming and applications. IEEE Expert 10, 5, 68–76.

YANGIYA, M. 1995. Efficient genetic programming based on binary decision diagrams. In 1995 IEEE Conference on Evolutionary Computation. Vol. 1. Perth, Australia, 234–239.

YU, T. 2001. Hierachical processing for evolving recursive and modular programs using higher order functions and lambda abstractions. Genetic Programming and Evol. Mach. 2, 4, 345–380.

ZHOU, A. 2003. Enhance emerging market stock selection. In Genetic Programming Theory and Practise, R. L. Riolo and B. Worzel, Eds. Kluwer, Chapter 18, 291–302.

A. MORE INFORMATION ON GENETIC PROGRAMMING

—`http://www.genetic-programming.org/`

—`http://en.wikipedia.org/wiki/Genetic_programming`

—`http://dbkgroup.org/gp_home.htm` Bioanalytical Sciences Group

—`http://evonet.lri.fr/TikiWiki/tiki-index.php?page=EvoBIO` EvoBIO conference

—GP Bibliography `http://www.cs.bham.ac.uk/~wbl/biblio/`

—`http://surf.de.uu.net/encore/` Encore, the electronic appendix to the hitch-hiker's guide to evolutionary computation.