# Multi-Chromosomal Genetic Programming

Rachel Cavill
University of York
York,UK
rc145@ohm.york.ac.uk

Steve Smith
University of York
York,UK
sls5@ohm.york.ac.uk

Andy Tyrrell
University of York
York,UK
amt@ohm.york.ac.uk

## ABSTRACT

This paper introduces an evolutionary algorithm which uses multiple chromosomes to evolve solutions to a symbolic regression problem. Inspiration for this algorithm is provided by the existence of multiple chromosomes in natural evolution, particularly in plants. A multi-chromosomal system usually requires a dominance system and subsequently dominance in nature and in previous artificial evolutionary systems has also been considered. An implementation of a multi-chromosomal system is presented with initial results which support the use of multi-chromosomal techniques in evolutionary algorithms.

## Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Global Optimization; I.2.2 [**Artificial Intelligence**]: Automatic Programming—*program synthesis*; I.2.8 [**Artificial Intelligence**]: Control methods and search

## General Terms

Algorithms Performance Design

## Keywords

Genetic Programming, representations, team evolution

## 1. INTRODUCTION

Many different representations have previously been proposed and some of these have proven to be very amenable to fast evolution of solutions to complex problems. However understanding what makes these representations particularly evolvable is still not well understood.

Sometimes, successful representations have imitated features seen in natural evolvable systems [15, 16, 18]. However, there are still many features which are seen in natural system whose value to artificial evolution has not been explored.


*GECCO'05*, June 25–29, 2005, Washington, DC, USA.


One such feature which is seen in more complex biological systems, is that of multiple chromosomes. This work explores the effects multiple chromosomes have in artificial evolution and their potential benefits for evolving more complex and effective programs.

This work considers an evolutionary algorithm which uses a biologically inspired, multi-chromosomal representation to evolve solutions to a symbolic regression problem.

## 2. GENETICS WITH MULTIPLE CHROMOSOMES

Genetics with multiple chromosomes differs in a number of respects from the artificial genetics, which is commonly used in artificial systems. The crossover operator (which takes multiple chromosomes and combines them in a meaningful way) and the need for a dominance system (to decide which of the genes on the multiple chromosomes applies) are two such major differences. The question we ask here is whether these mechanisms can help our artificial systems?

The following review of biological representations used in evolution has been compiled mainly using a variety of commonly available biological sources [6, 13, 22].

### 2.1 Biological Representations

In biology, there are two main categories of creatures whose genetic makeup differs considerably, the prokaryotes (mainly bacteria), which rely on a single circular chromosome, have different recombination operators due to the different shape of the chromosome and the fact that they reproduce by splitting a single cell, so no sex is involved.

Eurokaryotes (all creatures which are not prokaryotes), however all have multiple chromosomes and even multiple copies of each chromosome. This also means that they need a dominance system of some kind to 'decide' which of the genes, from these multiple chromosomes applies.

### 2.2 How many Chromosomes?

There is still an incomplete understanding in biology as to why some creatures have more chromosomes than others and why they have varying numbers of copies of each chromosome. Humans have 23 pairs of chromosomes and therefore since the chromosomes are paired, humans are termed *diploid* (as are all mammals). However, most plants and some reptiles and amphibians have more than two copies of each of their chromosomes, which makes them *polyploid*. Wheat for example has six copies of each of its seven chromosomes. Table 1 shows the differing numbers of chromosomes in a number of different plant species [1].

| Plant species | Number of different chromosomes | Copies of each | Total |
|---|---|---|---|
| Potato | 12 | 4 | 48 |
| Peanut | 10 | 4 | 40 |
| Sweet Potato | 15 | 6 | 90 |
| Tobacco | 12 | 4 | 48 |
| Wheat | 7 | 6 | 42 |
| Oats | 7 | 6 | 42 |
| Strawberry | 7 | 8 | 56 |
| Sugar Cane | 10 | 8 | 80 |

**Table 1: Plant species and their Chromosomes**

The occurrence of multiple copies of chromosomes appears to have come about through *doubling events* in the genetic history of the plants [30]. There are two types of doubling events and each can occur naturally as well as being artificially induced. The first type of event is a doubling within a single species - so a tomato species which previously had two copies of each chromosome (diploid), doubles its chromosomes and has four copies of each chromosome (tetraploid). A species which has undergone this type of doubling event is called autopolyploid.

The other type of doubling is when two different species combine their chromosomes to form a new species which has all the chromosomes from both species. This is called allopolyploidy of which wheat is a natural example [30].

Allopolyploidy is thought to be particularly important in the creation of new plant species and it is believed that all plants have undergone these doubling events at some point in their genetic history, although we can only detect the most recent events [30].

## 2.3 Crossover with multiple chromosomes

Crossover with multiple chromosomes can be seen as a multi-step process. To form the sex cells, which will merge and develop into a new individual, a crossover operation must be performed in the current cell. Pairs of identical chromosomes are chosen and match up so that like parts are aligned. Crossover points are chosen and the sections of the chromosome which are between these crossover points are swapped from one chromosome to the other. In the next stage, the altered chromosomes are shuffled with respect to their grandparents and the identical pairs separate to go into separate sex cells (see figure 1).

When polyploidal species pair off their chromosomes in preparation for crossover, in some species the pairings always take place in an identical manner, so that, if repeated, the same pairs would always be formed, in other species there is an element of randomness in how the chromosomes pair, and the same pairings would not be formed, if the pairing stage were to be repeated.

## 2.4 Dominance with multiple chromosomes

For diploid individuals, dominance can sometimes work completely, so there will be two possible genes, one dominant and one recessive. Any individual which has one or more dominant genes shows the dominant trait. Only individuals with both genes recessively set will show the recessive trait (see table 2).

However, not all genes work like this, even within diploid individuals. Other common models include; *co-dominance* where both genes are equally dominant and individuals with

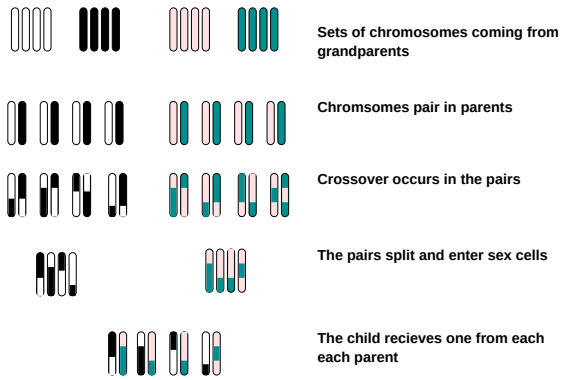


**Figure 1: A diagram showing how genes are passed down on the chromosomes from grandparents to grandchildren.**

| | X Dominant gene | x Recessive gene |
|---|---|---|
| X Dominant gene | XX Dominant trait | Xx Dominant trait |
| x Recessive gene | xX Dominant trait | xx Recessive trait |

**Table 2: The basic gene dominance system for humans**

one of each show a trait halfway between the two. Examples of this include white flower and red flower genes, mixing to give pink flowers, and the human blood group AB which is caused by individuals having one gene for blood group A and one for group B. Another common way dominance can work is through an *allele-dosage effect*: this is similar to co-dominance, but the dominant gene causes the trait to be 'more' than the recessive one in some respect and if the individual has two copies of the dominant gene the trait will be expressed even 'more'. Examples of the allele-dosage effect are often seen in fruit or plant size[30].

The allele-dosage effect can be seen in polyploidal species; only in these cases the individual with all dominant genes will have the trait expressed 'even more' than the individual with some dominant genes, which will itself have it expressed 'more' than the individual with all recessive genes (see figure 2).

Often the picture gets even more complicated than this. A gene may have more than two possible settings (more alleles) or by more than one gene affecting a trait, or even by one gene affecting more than one trait. Thus the effects of gene expression in polyploids is still very much under investigation in biology [30].

## 2.5 Summary

This section has covered how having multiple chromosomes is common in natural systems and we have considered the way in which crossover works in two stages to mix up the genes on each chromosome (with the genes from the other chromosome with which it pairs) and then the chromosomes get mixed up and combined with another set to create a child. We have considered dominance can work when the individual has two copies of each chromosome and how this can be extended to individuals who have more copies of each chromosome.

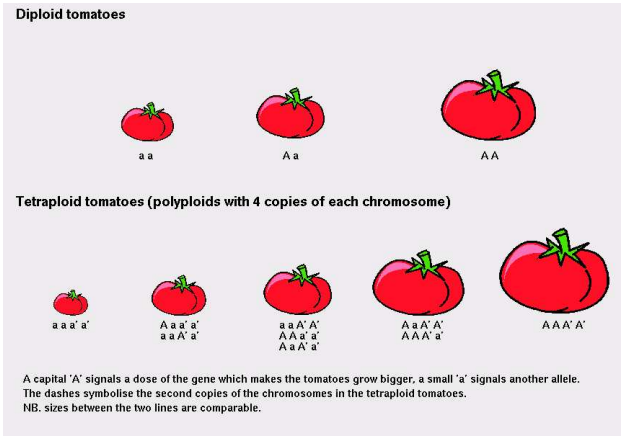


**Figure 2: A diagram of how fruit size *might* be affected if a variety of tomatoes became polyploid. Since tomato fruit size follows an allele-dosage affect it is not unreasonable to guess that this might be the outcome of a doubling in chromosomes.**

# 3. RELATED WORK

Although there has only been a limited amount of research undertaken into using diploidy and dominance in artificial evolution, some work on genetic algorithms and genetic programming which uses these features is described below.

Other related work considers evolving teams - since a team when evolved as a single entity can be viewed as a collection of chromosomes.

## 3.1 Multi-Chromosomal Evolutionary Algorithms

There are several examples of genetic algorithms (GAs) which have described themselves as multi-chromosomal. [10, 21, 11]. Hinterding [10], used a representation which had different genes translating into different types used different operators on these different types which were grouped into three chromosomes. Another approach [21], used multiple chromosomes to partition the search space and hence, unlike the other multi-chromosomal representations where the differing chromosomes represent different things and have different structures, this approach has identically structured chromosomes. The parameters of the functions to be solved were evenly distributed across varying numbers of chromosomes. They found that the multiple chromosomes produced improved results. These multi-chromosomal representations for GAs have been applied to problems such as pallet loading and cutting stock [10, 21, 11].

Going a stage further than these works, towards biologically plausible multi-chromosomality is the work by Hillis [9]. This early work not only uses multiple chromosomes, but also a diploid system and a bio-inspired crossover operator much like the one described here. Each gene in each chromosome codes for an exchange, a pair of numbers in the list to be sorted which will be compared and then exchanged if needed. They use the diploidy inherent in their system to generate differing numbers of exchanges to alter the size of the sorting network they are evolving. If a pair of chromosomes differs at a particular point then both exchanges are expressed, otherwise only one exchange is expressed.

## 3.2 Dominance in Genetic Algorithms

Most of the work done experimenting with dominance in artificial evolution is in the field of genetic algorithms [5, 14, 25, 23].

The advantages of adding dominance to GAs are disputed, but evidence seems to show that on non-stationary problems, particularly those that flip between two solutions, GAs with some dominance schemes have a particular advantage [5, 14].

## 3.3 Dominance in Genetic Programming

Even when using a single chromosome in genetic programming, implementing a dominance system has been attempted [31]. In this work crossover is done by combining two subtrees to make one, then adding this to both parents to make two children. The nodes which get to be expressed after the combination process are those with the highest dominance values. Dominance values are increased on nodes when a child produces a better result than its parents. However, they found there were problems integrating this type of system with the standard genetic programming representation.

## 3.4 Evolving Teams

Perhaps the closest work to this research has been in the field of evolving teams of agents to do some task. Traditionally the agents would have been evolved to perform some team oriented task, for instance a predator prey simulation with multiple predators who have to co-ordinate to catch the prey [17, 8, 7]. More recently techniques which were first applied to these type of problems have been adapted so that they can be used to solve other more types of problems Soule, who uses team evolution to solve symbolic regression problems, including the 7-bit parity problem [27, 26].

One problem when evolving a team is whether to evolve the team as a whole (ie. have a population of teams which are evolving), or you evolve the members of the team in the population and then split the population up into teams when testing for fitness. Clearly the first of these options is much more strongly related to the work presented in this paper, because it is only when you evolve a team as a whole that you will get something which can be viewed as having multiple chromosomes evolving.

When evolving a team as a whole, you have a second choice regarding the crossover operator. You could allow crossover between any parts of the team with any parts of any other team, this would encourage homogeneous teams to evolve. However, if you restrict crossover so that it only crosses the first member of a team with the first member of other teams, and the order of team members is preserved throughout the run, then heterogenous teams can easily emerge. This allows specialisation and the work becomes even more closely related to what is presented here.

Despite the similarities between this research and team evolution research there are still some differences. The system presented here has multiple copies of each of a set of chromosomes. (eg. 4 copies of 4 chromosomes totalling 16 chromosomes). This *bio-inspired* approach will therefore encompass the benefits of evolving teams, in a broader setting.

Research with evolving teams has produced good results with researchers finding that bloat in the individual team members is reduced [27] significantly when compared to a population of individuals evolved for the same problem.

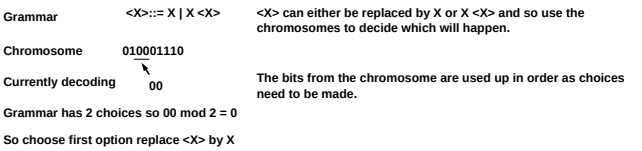


**Figure 3: A diagram showing a simple GE translation process, taking a binary string and using a set grammar to convert it into a statement which obeys that grammar.**

# 4. MULTI-CHROMOSOMAL GENETIC PROGRAMMING

Multi-Chromosomal genetic programming is a new technique which aims to improve the ability to evolve complex structures by splitting the representation up into blocks. Having noted that many complex structures in nature use multiple chromosomes, the aim of this work is to explore their usage within genetic programming.

## 4.1 The Technique

The chromosomes in the multi-chromosomal genetic programming system are all binary chromosomes which are translated into Prolog functions using a grammatical evolution type translation process. Each chromosome translates into a Prolog clause and the clauses may call each other and themselves, allowing for recursion.

Grammatical Evolution (GE) [24] is an evolutionary technique which uses a binary chromosome to select which path the definition of the program should take. The possible programs are defined using a Backus-Naur form grammar and at every point where a decision has to be made about which substitution to make in the forming of the program then the next chromosome is used to decide. (see figure 3 for a simple example).

GE has proved successful at evolving code in different languages and at solving a wide variety of problems from business problems, such as whether the decisions of managers affect the prospects of the business [4] to modeling complex gene interactions [19].

Since the GE translation process has proved to be a successful translation process for these many varying applications, it was chosen for this work.

Prolog [29] was chosen as a language for the programs to be evolved into because it could easily cope with over and under specification of 'functions'. A prolog clause is designed to cover one case of a problem, and it may take many clauses each needing the same inputs and outputs to cover all cases (see figure 4), hence prolog is designed to try each possible clause in turn (in the order they are listed in the program code). Multi-chromosomal genetic programming takes advantage of this feature by using a dominance system, consisting of a real number attached to each chromosome, which may itself be mutated and so will evolve to order the clauses. This means that multiple definitions of the same type are encouraged rather than being problematic as they would be in most other programming languages. Under specification, too, is not a problem for a program evolved in Prolog. If a clause tries to call a clause which doesn't exist it just returns a fail. This feature is not used in the current version of the system, since all programs are constrained to only call clauses which do exist.

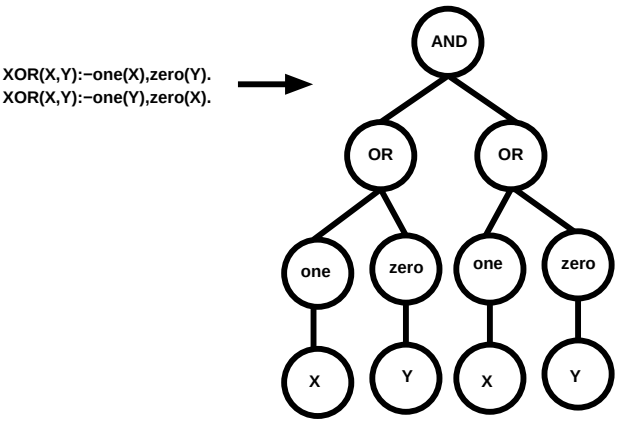


**Figure 4: A diagram of how multiple prolog clauses fit together using the xor function as an example and showing the parse tree that they would generate if they were represented in that fashion.**

| Parameter | Default Value |
|---|---|
| Population size | 100 |
| Mutation bit rate | 1/500 |
| Min size of chromosome at start of run | 120 |
| Generations | 100 |
| Tournament size | 4 |
| Number of different chromosomes | 4 |
| Number of copies of each chromosome | 4 |
| Number of elite | 3 |
| Maximum number of crossover points | 3 |

**Table 3: Current Parameter Values**

## 4.2 Current settings and the problem

Currently the system uses four copies of each of four chromosomes and then orders the evolved prolog clauses according to a dominance figure. This dominance figure is attached to each chromosome and also evolved. A full crossover including crossing over the binary strings (not just shuffling the chromosomes) is done 50% of the time and a bit mutation rate of 1 in 500 was used throughout. For a full listing of the default settings see table 3. With these settings a full run takes around 1 hour on a modern PC.

The problem which is tackled here is one of the earliest symbolic regression problems which Koza solved ([12] chapter 7) to demonstrate the power and scope of his system. It was chosen because it was felt that comparing early versions of one technique to early versions of another would be the fairest method. The function to be matched is $x^4 + x^3 + x^2 + x$. The function set used includes plus, minus, times, protected division, sine, cosine, exponential and protected logarithm. There are twenty test points which are spread out equally over the range [-1,1] and the fitness function takes the total absolute difference of the evolved answers and the real answers. Since each program is assigned a small amount of time to run (to protect against any infinite loops which may be evolved) there are additional penalties added to the fitness of any individual which does not complete all the calculations in the time. Those individuals which do nothing are assigned a *dead* status and cannot be picked as parents.

This problem has also been used more recently by researchers using Grammatical Evolution [20, 2]. This is particularly pertinent, since this work uses a similar translation process so comparisons with this work should be more meaningful.

## 5. RESULTS

The results are averaged over 20 runs and the ideal fitness score is zero, which means that the target function, $x^4+x^3+x^2+x$, has been matched at all points.

The graph in figure 6 shows how throughout the run the best fitness scores continue to decrease steadily towards the ideal fitness of zero. The figure 5 also shows the median and mean and worst fitness scores for the population, as averaged over 20 runs. Although the mean fitness scores of the population do not appear to be following a trend towards the ideal fitness score, the median obviously is tending towards this ideal. This is because a few bad individuals can easily throw the mean off the general trend.

These results show that evolution is taking place in the system and that both the best individual and the population as a whole (as shown by the median) are improving as might be expected.

Figure 7 shows a comparison of the system set to use a single chromosome compared with the standard settings used elsewhere in this paper. It also includes the results for when a single chromosome was initially set to 16 times the normal length of a chromosome (so that it has the same amount of genetic material as the multiple chromosomes) and the results for when these longer chromosomes are split into 16 equally size parts for fitness evaluations, but are evolved as a single string. The graph shows that not only is the initial population of the multi-chromosomal system far superior to the initial population when only one chromosome is used, but that evolution throughout the run proceeds much better when multiple chromosomes are used, with the gradient of descent being much steeper for multiple chromosomes and the gap widening throughout the run. The single chromosome which was split up into 16 parts for fitness evaluations initially performs well, like the multiple chromosomes, but then is unable to evolve further. Overall these tests show that it is the multiple chromosomes and the way they are evolved, not just having more genetic material which is causing the improvement in performance.

To test whether evolution continued after 100 generations, or whether it tailed off, a series of longer runs were performed, allowing 1000 generations to pass. this demonstrated that evolution is still going strong long after the previous limit of 100 generations had ended.

These results are compared with those obtained by other methods in section 6.

## 6. DISCUSSION

With any new system it is always important to compare the results with those of other equivalent methods to show whether the system has any added value over current practice. This comparison of results is slightly hindered by the fact that other methods have not presented thorough results for this particular problem in the literature. For example Koza [12] only reports one run for this problem. In that run a fully correct individual was found in generation 34. However, it is interesting to note that the mean scores from

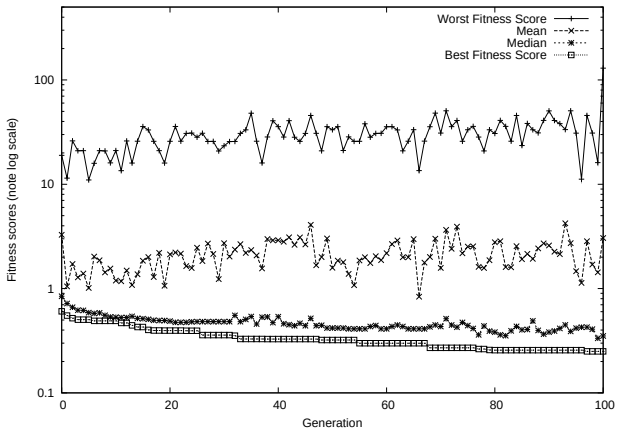


**Figure 5: A graph showing how the best, worst, mean and median fitness. All results are averaged over 20 runs**

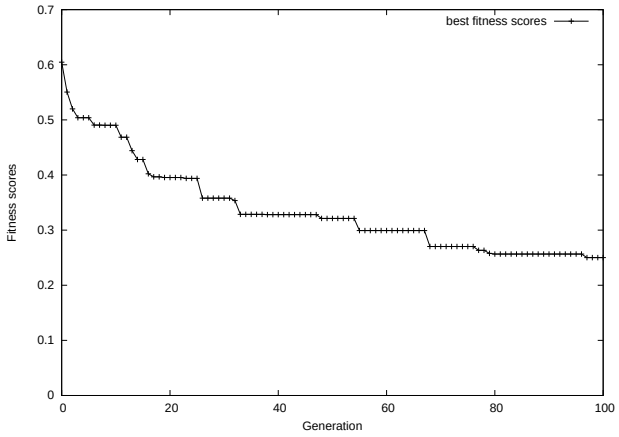


**Figure 6: The best fitness scores from figure 5 enlarged without a log scale**

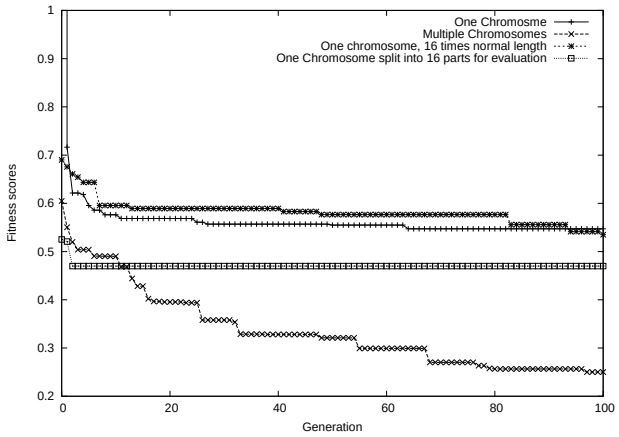


**Figure 7: Comparing using 16 chromosomes per individual with using just one, all figures averaged over 20 runs**

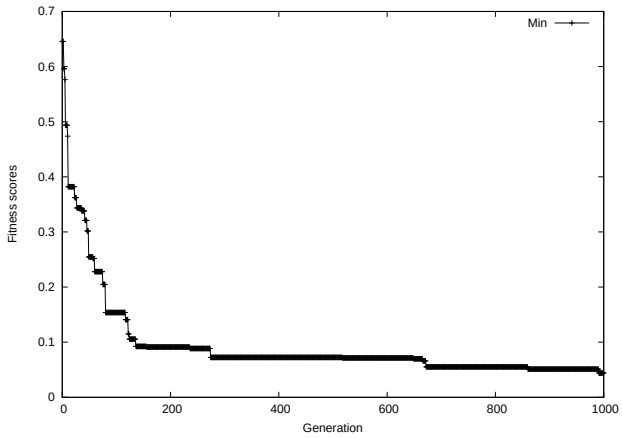


**Figure 8: The best individuals fitness scores from longer runs (currently averaged over only 7 runs due to time constraints)**

Koza's work are considerably worse than those obtained by the multi-chromosomal system, despite the mean scores in this system being raised by some high scoring individuals who were penalized for overrunning the time and the median scores, providing a much more indicative measure of the performance of the population as a whole. It is also of interest that for the multi-chromosomal system, the fitness results steadily decrease throughout the run, whereas Koza's results seem to decrease more slowly and then show a sharp drop from a fitness of around one in generation 33 to a fitness of zero in generation 34.

This problem has also been applied to Grammatical evolution [24] where it was used as an initial proof-of-concept. However, it is only reported that grammatical evolution can consistently find perfect answers.

One crucial factor which has been problematic in many genetic programming systems is the problem of bloat [3, 28], the uncontrollable growth of evolved solutions. Therefore some initial analysis of how much the representation suffers from bloat has been done. The number of parts to the 16 evolved functions of each individual was counted. The counts were done both for the initial random population and for the final evolved population, after 100 generations. The average per individual at the start of the run was 37.85 and by the end of the run this had risen slightly to 43.32. However, given the standard deviations in both cases were around eight, this is a change of less than one standard deviation and therefore this low level of growth over 100 generations, shows that the representation has a strong parsimony pressure which serves as some protection against the phenomenon of bloat.

## 7. FUTURE WORK

The current work demonstrates that it is possible to evolve solutions to a symbolic regression problem using a multi-chromosomal representation. Further investigations are currently being undertaken into how the system reacts with different settings and on different problems.

Some further analysis of bloat, measuring the lengths of the binary chromosomes, and analysing the size of programs at other stages in the evolutionary run, may be insightful.

There is a need to investigate the parts of the system individually to break down the individual effects and determine which are the most beneficial to artificial evolution. Other experiments could allow only multiple copies of a single chromosome, or the pairing patterns of chromosomes could be investigated (does it help to pre-specify which chromosomes will pair, or is it better to allow any 'like' chromosomes to pair?).

## 8. SUMMARY

This paper has described a new representation for evolutionary algorithms which uses multiple chromosomes to evolve and its underlying biological inspiration. It has been shown that evolution with multiple chromosomes, within this new representation, is advantageous over a conventional single chromosome. Initial results therefore suggest that multiple chromosomes are a promising approach and warrant further investigation.

## 9. REFERENCES

[1] Botany online: Classic genetics - chromosomal numbers - autoploploics - somatic polyploidy. University of Hamburg's webapges http://www.biologie.uni-hamburg.de/b-online/e12/12a.htm.

[2] R. M. A. Azad and C. Ryan. Structural emergence with order independent representations. In E. Cant'u-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, pages 1626–1638, Berlin, 2003. Springer-Verlag.

[3] W. Banzhaf and W. B. Langdon. Some considerations on the reasons for bloat. *Genetic Programming and Evolvable Machines*, 3:81–91, 2002.

[4] A. Brabazon, M. O'Neill, C. Ryan, and R. Matthews. Evolving classifiers to model the relationship between strategy and corporate performance using grammatical evolution. *Lecture Notes in Computer Science*, 2278:103–112, 2002.

[5] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates, Publishers, 1987.

[6] A. Griffiths. *Introduction to Genetic Analysis*. WH Freeman, 2000.

[7] T. Haynes and S. Sen. Crossover operators for evolving A team. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 162–167, Stanford University, CA, USA, 13-16 1997. Morgan Kaufmann.

[8] T. Haynes, S. Sen, D. Schoenefeld, and R. Wainwright. Evolving a team. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, Cambridge, MA, 1995. AAAI.

[9] D. W. Hillis. Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D*, 42:228–234, 1990.

[10] R. Hinterding. Self-adaptation using multichromosomes. Technical Report, 1997.

[11] R. Hinterding and K. Juliff. A genetic algorithm for stock cutting: An exploration of mapping schemes. Technical Report, 1993.

[12] J. R. Koza. *Genetic Programming*. MIT press, 1992.

[13] E. Lawrence. *Henderson's dictionary of biological terms; 11th edition*. Addison Wesley Longman Limited, 1995.

[14] J. Lewis, E. Hart, and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In A. E. Eiben, T. B"ack, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 139–148, Berlin, 1998. Springer. Lecture Notes in Computer Science 1498.

[15] M. A. Lones and A. M. Tyrrell. Enzyme genetic programming. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1183–1190, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.

[16] S. Luke, S. Hamahashi, and H. Kitano. "genetic" programming. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, pages 1098–1105, San Francisco, CA, 1999. Morgan Kaufmann.

[17] S. Luke and L. Spector. Evolving teamwork and coordination with genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Stanford University, CA, USA, 28–31 1996. MIT Press.

[18] S. Margetts and A. J. Jones. An adaptive mapping for developmental genetic programming. In J. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tetamanzi, and W. B. Langdon, editors, *Proceedings of the Fourth European Conference on Genetic Programming (EuroGP-2001)*, volume 2038 of *LNCS*, pages 97–107, Lake Como, Italy, 2001. Springer Verlag.

[19] J. H. Moore and L. W. Hahn. Petri net modeling of high-order genetic systems using grammatical evolution. *BioSystems*, 72:177–186, 2003.

[20] M. O'Neill and C. Ryan. Under the hood of grammatical evolution. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, pages 1143–1148, San Francisco, CA, 1999. Morgan Kaufmann.

[21] H. J. Pierrot and R. Hinterding. Using multi-chromosomes to solve a simple mixed integer problem. In *Australian Joint Conference on Artificial Intelligence*, pages 137–146, 1997.

[22] Roberts. *Biology: A functional Approach*. Nelson, 1982.

[23] C. Ryan. The degree of oneness. In *1st Online Workshop on Soft Computing*, pages 100–105, Nagoya, Japan, 1996.

[24] C. Ryan, J. J. Collins, and M. O Neill. Grammatical evolution: Evolving programs for an arbitrary language. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391, pages 83–95, Paris, 14-15 1998. Springer-Verlag.

[25] R. E. Smith and D. E. Goldberg. Diploidy and dominance in artificial genetic search. *Complex Systems*, 6(3):251–286, 1992.

[26] T. Soule. Voting teams: A cooperative approach to non-typical problems using genetic programming. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, pages 916–922, San Francisco, CA, 1999. Morgan Kaufmann.

[27] T. Soule. Heterogeneity and specialization in evolving teams. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 778–785, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[28] T. Soule and R. B. Heckendorn. An analysis of the causes of code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 3:283–309, 2002.

[29] L. Sterling and E. Shapiro. *The Art of Prolog*. MIT Press, 1986.

[30] J. A. B. Thomas C Osborn, J Chris Pires, D. L. Auger, Z. J. Chen, H.-S. Lee, L. Comai, A. Madlung, R. D. V. Colot, and R. A. Martienssen. Understanding mechanisms of novel gene expression in polyploids. *Trends in Genetics*, 19(3):141–147, 2003.

[31] K. Vekaria and C. Clack. Haploid genetic programming with dominance. Technical Report RN/97/121, 1997.