Finding Needles in Haystacks is Harder with Neutrality.

M. Collins CISA
Edinburgh University
Scotland
mcc@dcs.ed.ac.uk

ABSTRACT

This research presents an analysis of the reported successes of the Cartesian Genetic Programming method on a simplified form of the Boolean parity problem. We show the method of sampling used by the CGP is significantly less effective at locating solutions than the solution density of the corresponding formula space would warrant.

We present results indicating that the loss of performance is caused by the sampling bias of the CGP, due to the neutrality friendly representation. We implement a simple intron free random sampling algorithm which performs considerably better on the same problem and then explain how such performance is possible.

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: Automatic Programming; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms

Keywords

Reduced Boolean Parity, Cartesian Genetic Programming, Search Space

1. INTRODUCTION

This paper investigates the expected and actual success rates of the Cartesian Genetic Programming (CGP) algorithm as reported by Yu and Miller in [7]. The CGP system has gained attention following its performance on certain types of Boolean parity problems, a classic test problem in both Evolutionary Algorithm and Evolutionary Hardware research.

We compare the reported performance of the CGP algorithm on the Boolean parity problem to the performance of a CGP style random walk algorithm and a random resampling algorithm, using the same parameters and representation. We were unable to repeat the same performance as reported for the CGP, but are able to show that all CGP based sampling methods - including the unrepeated reported performance - are underperforming in terms of solutions per sample when the solution density is considered.

We also implement and test a trivial random sampling algorithm which uses a fully expressed genotype. We show that this algorithm considerably exceeds expectations unexpectedly performing better than random sampling. An analysis of how simply removing introns from the representation can produce such a result is provided.

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

2. PREVIOUS WORK

Previous work by Langdon and Poli [3] into the structure of solutions to Boolean problems, including a special form of the parity problem using a reduced operator set of only the XOR EQ operators indicated the solution density in the space of formulas tended to a limit. Hereafter this special form of the Boolean parity problem which is solved using only EQ and/or the XOR operator will be referred to as the 'reduced Boolean parity problem' to distinguish it from the version with unrestricted logical operators.

Yu and Miller used the reduced Boolean parity problem as a test case for the CGP algorithm. CGP used a representation which supports introns, the belief being that the introns and the 'neutrality' which they permit could be exploited by the search algorithm.

Previous work [1, 2] has provided and empirically proved the accuracy of a method for counting the number of solutions to the reduced Boolean parity problem in the space of all possible formulas. This gives a method for calculating the number of solutions which are in existence in the space sampled by the algorithms. This work supported the findings of Langdon and Poli and gave exact results for the solution density and the limit.

3. REPORTED PERFORMANCE

The CGP has performance reported for the 5 8 10 and 12 parity problems. We will concentrate on the 12 parity problem, since this is **a)** the hardest of the reduced Boolean parity problems for which CGP results are reported, **b)** Yu and Miller reported the failure of random sampling on this problem see ([7] page 5, table 1), and, **c)** Yu and Miller obtained an outstanding >55 % success rate from a hundred runs, each of a mere 10,000 iterations. We use only reported results for comparison in this investigation.

4. REPRESENTATION ISSUES

In the Boolean parity problem the input alphabet I of size $|I|$ is a set of distinct Boolean values $\{I_1, ..., I_{|I|}\}$, corresponding to the parity of the problem.

4.1 The space of all possible formulas

In the reduced Boolean p-parity problem, the input alphabet is p elements, and the function set is either {XOR,EQ} or simply only {EQ}. The number of possible arrangements using from 1 up to F_{Max} functions is:

$$\sum_{i=1}^{F_{Max}} 2^i p^{i+1} \quad (1)$$

Where an example formula constructed using 2 functions and an alphabet of 3 inputs is: I_1 EQ I_2 EQ I_3 and is a solution to the 3 parity problem. Another formula but this time constructed with 3 functions, I_1 EQ I_2 EQ I_3 EQ I_1 is not a solution to the 3 parity problem.

4.2 The number of solutions

The number of formulas which are solutions to the reduced Boolean p parity problem for all possible functions referencing up to a maximum of n inputs is shown in figure 1. Please see earlier work in [1, 2] for an explanation of the derivation of this equation and empirical evidence of its accuracy.

4.3 The CGP representation

In the reported CGP representation the genotypes are composed of sequences of integer triples, each of which represents a unit equivalent to a reprogramable gate or cell. The first integer represents the gate type, though the reported work on the 12 parity problem only used one gate type. The second and the third integers represent the connections to either one of the 12 parity inputs or the outputs of previous gates. The output of the gate is then the result of applying the operation represented by the gate type to the inputs referenced by the gate. The values of the input references are limited to enforce a feedforward connectivity which with a fixed interpretation order removes the requirement for a clock and prevents looping. The final output is determined by selecting one of the gates and tracing back the connections to previous gates and inputs, which are then evaluated in the logical order.

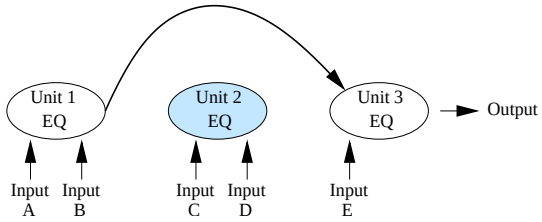


Figure 2: Interpretation of a CGP genotype is dependent upon the unit selected to generate the output and the subsequent linkage of the units. Where units are not linked to the output unit (here marked with colour) they are introns and candidates for neutral mutations.

Figure 2 shows a simple CGP arrangement corresponding to the genotype (EQ,A,B)(EQ,C,D)(EQ,1,E) (interpreted from unit 3) where letters represent references to the p inputs and numbers represent references to the output of previous units. The actual active expression of this genotype is A EQ B EQ E.

The significant feature of the CGP representation is that the genotype implicitly supports 'neutral' mutations. Introns in CGP are sections of the genotype which are left out of the interpretation due to simply not being included in the chain of references from the chosen output gate. Mutation of the introned code has no effect on the expression of the genotype or the phenotype and is termed 'neutral'. Neutral movement in the genotype is hypothesised to aid search by allowing small mutations to accumulate to create large and complex genotype changes. These changes can then be easily brought into and removed from the phenotype by relatively small mutations in referencing code.

Since the CGP representation has introns as an enevitable part of its representation, it has a bias towards representing smaller solutions more frequently than a representation in which all the code was expressed: There is no one to one mapping from the formula which are represented by CGP solutions and the space of possible formula. The question is, does this help or hinder? As a benchmark for researching neutrality the reduced Boolean parity problem is actually a strange choice - due to the fitness function, all movement except that which finds a solution is score neutral. Differences in performance between the CGP and other tactics on the same landscape must then be a function of the effect of neutrality on the sampling strategy, and not due to permitting score neutral movement.

5. THE CONTEXT OF THIS INVESTIGATION

The aim of this investigation is to determine if the CGP representation is better or worse at generating solutions than one would have a right to expect from a given solution density.

Previous work [1, 2] has provided calculations for the number of solutions present in the space of possible functions. For Boolean formulas using between 1 and 100 operations, the solution density of the reduced Boolean 12 parity problem space is 0.0019531 percent for two operator types, and 0.003756 percent for one operator type. Since the reduced Boolean search space is devoid of fitness gradient clues, and consequently search is unguided, this represents the best expected performance for an algorithm which samples the space without bias.

In [7] Yu and Miller report CGP has a peak success rate of over 55% on the reduced Boolean 12 parity problem. The average success rate is approximately 45% for the optimal combination of parameters. These results are achieved using the CGP algorithm over 10,000 iterations, which is a sample of at most 40,000 points.

A search using one operator type (EQ), sampling evenly from the search space of possible formulas until finding a solution, performing 40,000 samples has an expected success rate of $1 - (1 - 0.00003756)^{40000} = 0.778$. This contrasts with the empirically obtained expectation of just over 0.55 which was reported for the CGP. The difference in performance is then a consequence of the mapping imposed by the CGP implicitly neutral representation and the CGP search tactics.

$$\sum_{i=p}^{n} 2^{(n-1)} \sum^{\forall \pi} \left\{ \frac{n!}{\prod_{a=1}^{|\pi|} (2\pi_a + 1)!} \prod_{b=1}^{|\pi|} \binom{p - \sum_{c=0}^{b-1} \sum_{b=1}^{|\pi|} (\pi_b = s)}{\sum_{d=1}^{|\pi|} (\pi_d = s)} \right\}$$

Where n is the maximum number of inputs, p is the parity and π represents the set of integer partitions of $(n-p)/2$.

Figure 1: The number of solutions in formula space of the reduced Boolean parity problem

6. THE EFFECT OF AN IMPLICITLY NEUTRAL REPRESENTATION

The CGP representation method encourages a simple graph structure, in which functional sequences are established by chaining outputs of earlier units to inputs of later ones. Inevitably some sections of the genotype are not interpreted, and it is these sections in which neutral mutation is expected to occur, though due to the scoring function of the reduced Boolean parity problem, all mutations except those which create solutions are score neutral. Using some of the genotype to represent introns shortens the expressions which can be composed for any given genotype length. This in turn has an interesting effect on the ability of CGP style representations to sample the reduced Boolean formula space effectively.

6.1 Candidate generation in CGP

Irrespective of the quantity of the genotype that is expressed, all CGP candidates have a genotype which is fully defined for all units. Assuming the use of a suitable random selection method, the following proceedure may be used to create a candidate solution to the reduced Boolean parity problem in the CGP representation:

```
set the output of the expression to:
   the output of a randomly selected unit.

for each unit, do
   set the type of the unit to:
        a randomly selected gate type
   set each of the input connections to:
        a randomly selected choice from all the
        previous gate outputs and the initial
        input lines
done
```

The CGP population strategy is a $(1+\lambda)$ evolution strategy with $\lambda = 4$ samples being created by mutating the sole parent. Samples replace the parent if certain criteria regarding active genotypic difference and score are met, (see [7, 8, 4, 5] for details). In the reduced Boolean parity problem, the score is always the same unless a solution is found. As reported by Yu and Miller in [7], relaxing the active genotype similarity constraint does not impair search performance.

As figure 3 shows, the default setting of the CGP algorithm favours shorter sequences than those which are capable of yeilding solutions to the 12 parity problem. Generating more offspring in each iteration does not smooth out the sampling. This sampling bias is more extraordinary when it is taken into account that there are vastly more lengthly formulas than there are short.

By increasing the length of the representation used, more promising areas of the formula space can be sampled by the CGP algorithm. Figures 4(a) and 4(b) show the distribution of samples as performed by; the CGP algorithm, and by random sampling of the 1000 unit CGP representation space.

Notice that no representation has a larger effective size than 150 units - a vast amount of the space is not sampled. Smoothing out the sampling of the representation space reduces the amount of samples which are wasted on sampling formulas less than 11 operators in length. This has a noticable effect on the success rate of the algorithm, which changes from approximately 20% to approximately 90% as the sampling method is changed from the CGP controlled random walk to independant random sampling of the CGP space.

Using the CGP representation and the CGP search strategy we have been unable to recreate the original results. The best results we obtained using the 100 unit CGP representation and the (1+4) CGP strategy over 10,000 iterations (40,000 samples) was 22% of runs resulting in a solution being located, and are frequently much lower.

6.2 Randomly sampling CGP representations

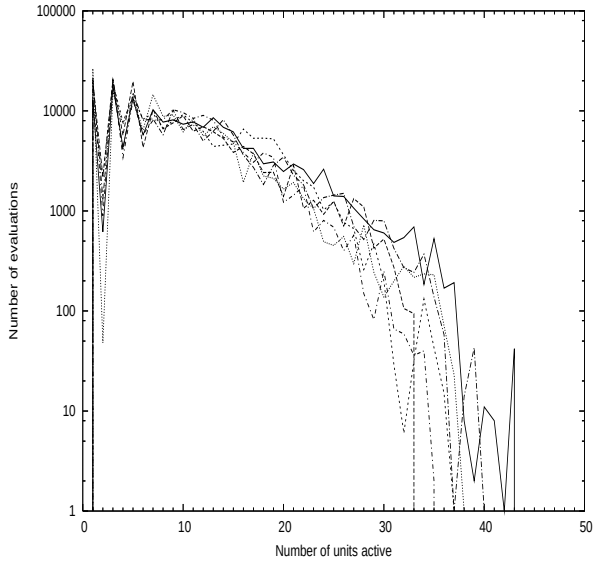
The significance of the inheritability of traits from the parent to effectiveness of the CGP search can be tested by comparing the performance of CGP to that of an algorithm which simply samples from the CGP representation space.

The random sampling of the CGP representation shares essentially the same function length distribution as the sampling by the CGP algorithm. Figure 5 shows the distribution of samples generated at random using the CGP representation. Comparing figures 3 and 5 shows the exact nature of the sampling - in terms of the proportions of representation lengths sampled is largely independant of the CGP search mechanism for representations of this length.

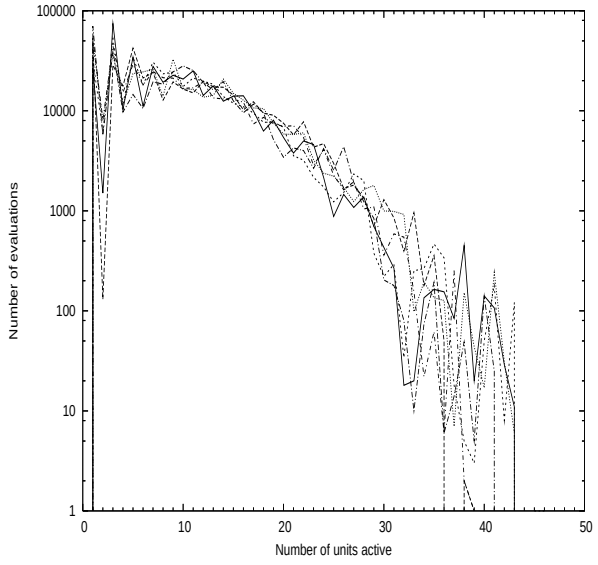
6.3 Randomly sampling non-CGP representations

Replacing the CGP representation with one that does not permit introns allows us to compare the effect of using a CGP representation with a representation which does not permit 'neutral' genotype changes. All changes to the genotype in such representation will be interpreted, though it does not necessarilly follow that they will alter the actual phenotype.

The following method generates candidates which sample from the space of possible reduced Boolean formulas. The chaining property, where the output of one unit is fed into the next, must be maintained if introns are to be avoided. This is achieved here by marking the reference immutable, which allows the similarity between the representation generation methods to be clearly seen. In practice this is most simply done by implicitly coding the structure and using the genotype to specify only the function type and the input reference(s) for each unit.

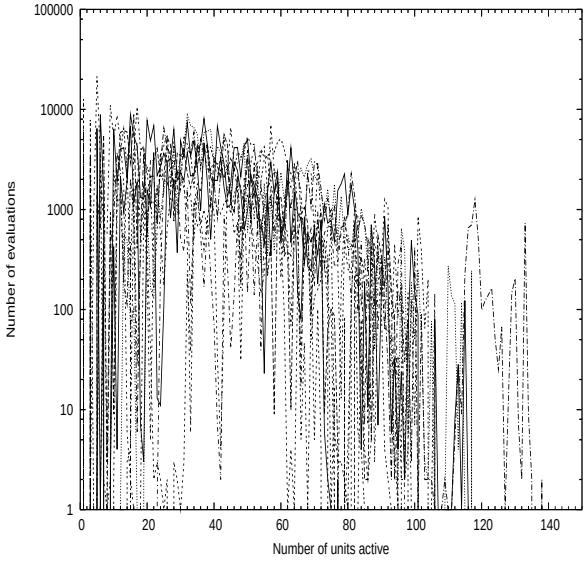


(a) 100 Units: 4 Children per iteration, : $(1 + 4\lambda$ strategy)

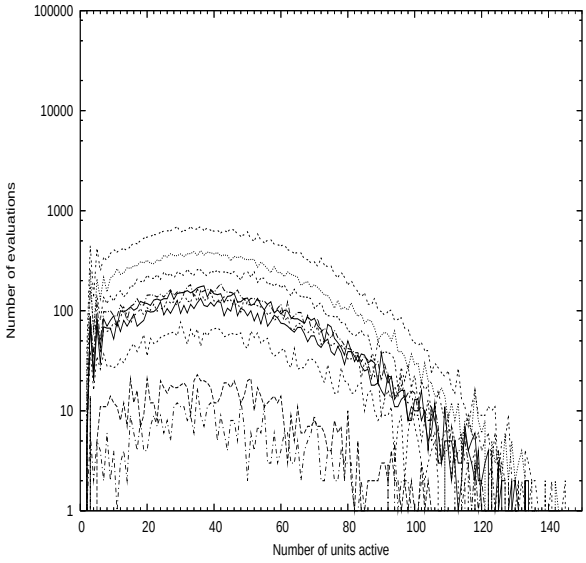


(b) 100 Units: 10 Children per iteration: $(1 + 10\lambda$ strategy)

Figure 3: The distribution of samples, in terms of the number of gates active, throughout 5 runs of the CGP algorithm. Note the Y axis log scale.

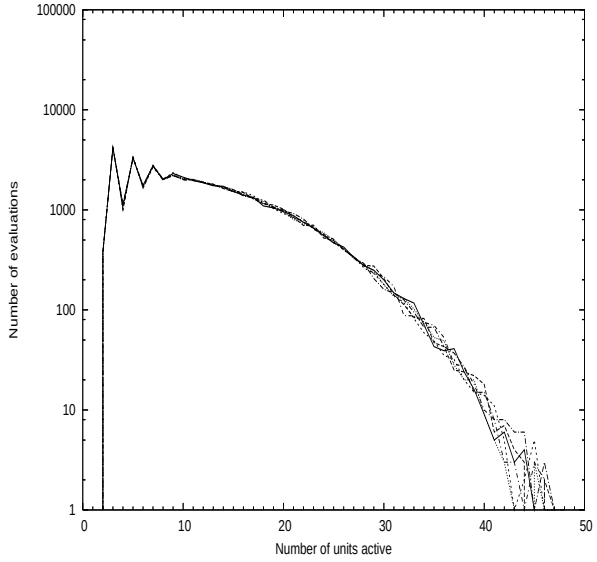


(a) 1000 Units: 4 Children per iteration, 20% success



(b) 1000 Units: random sampling of the CGP representation space, 90% success

Figure 4: The distribution of samples for representations using 1000 units, in terms of the number of gates active throughout 5 runs of the CGP algorithm. Results shown are a) the CGP sampling method and b) random sampling. Note the Y axis log scale.



(a) Randomly sampling 40,000 independent samples from the CGP representation space

Figure 5: The distribution of samples, in terms of the number of gates active, through 5 repetitions of 40,000 independent samples of the CGP representation space. Note the Y axis log scale.

```
select a random expression length L

set the output of the expression to:
   the output of the unit L.

for each unit up to L, do
   set the type of the unit to:
        a random gate type
   set one of the input connections of the unit to:
        the previous gate (make this immutable)
   set the other input connection to
        a randomly selected input line
done
```

Notice that this sampling method is also not 'fair' - it over samples short functions as well, however, it allocates approximately the same number of samples to each of the possible function lengths, and as a consequence has a much better success rate, we recorded 0.0204% success from a sample of 10 million trials. This is equivalent to expecting a solution to be located every 5000 trials.

6.4 Uniformly sampling the formula space

For formulas using up to 100 operators of one type the actual solution density is approximately 0.003756%. For each sample of 40,000 independently selected formulas this gives an expected solution occurence of 1.502 per 40000 samples. Thus an algorithm which samples uniformly from the space of formulas and stops when it finds a solution has an expectation of success of 0.778. The simple sampling strategy has an expected success rate of around 0.9997 under the same conditions.

The apparent 'overperformance' of the random sampling is entirely due to the distribution of the samples it uses, in particular the manner in which it avoids sampling the very largest formulas in favour of shorter formulas. Uniform sampling of the space would result in a large proportion of the test cases being drawn from the set of formulas of length 100 operators, since so much of the formula space is composed of these formulas. All formulas which reference an odd number of inputs (such as those with 100 operators) are not solutions to the reduced Boolean 12 parity problem. Hence the vast majority of the search space is occupied by formulas which are not solutions. Algorithms which are capable of sampling long formula lengths less than a uniform sampling method but still maintain sufficient sampling of all areas to detect solutions are likely to yeild better than random results.

7. CONCLUSION

One of the key features of the neutrality argument is that a representation which permits neutrality in genotype changes does not hinder nor worsen performance on the problems where neutrality is not of benefit. One of the findings of the Yu and Millar paper studying the reduced Boolean parity probem is that they found neutrality did not impair performance in those cases where it did not give improvement. The performance of the CGP algorithm on the reduced Boolean 12 parity problem seems to indicate that the asymmetry in the intron mappings does in fact cause damage to the ability of the algorithm to sample the space effectively.

In this work we have presented evidence that the CGP results as reported in [7] are actually worse than random. This serves to highlight that the effect of neutrality in evolution is not as benign as it might first appear, and reinforces the case for a full understanding of the test problems on which algorithms are evaluated.

8. ANTICIPATED RESULTS

Thanks to Julian Miller for his insightful response when shown a draft construction of the paper in November 2004. At the time of writing (14^{th} April 2005) we are awaiting his detailed response following a re-running of the CGP experiment set from [7].

9. REFERENCES

[1] M. Collins. *Counting Solutions in Reduced Boolean Parity* 2004. *GECCO 2004.*

[2] M. Collins. *Monte Carlo Sampling and Counting Solutions in Reduced Boolean Parity* 2004.*EDI-INF-RR-0240. http://www.inf.ed.ac.uk/publications/report/*

[3] W. Langdon. R. Poli. *Boolean Functions Fitness Spaces* 1997. University of Birmingham Technical Report CSRP-98-16.

[4] J. Miller. *An empirical study of the efficiency of learning boolean functions using a Cartesian Genetic Program ming approach* R. Poli et al (Eds.) Proceedings of the Third European Conference on Genetic Programming. 2000. pp. 121-132.

[5] J. Miller. *Cartesian Genetic Programming* R. poli et al (Eds.) Genetic Programming, Proceedings of EuroGP'2000

[6] J. Miller. *What Bloat? Cartesian Genetic Programming on Boolean problems* E. Goodman (Ed.) 2001 Genetic

and Evolutionary Computation Conference Late Breaking Papers, pp 295-302.

[7] T. Yu. J. Miller. *Finding Needles in Haystacks Is Not Hard with Neutrality* J. Foster et al (Eds.) EuroGP 2002, LNCS 2278, pp. 13-25.

[8] T. Yu. J. Miller. *Neutrality and the Evolvabilty of Boolean Function Landscapes* Proceedings of the Fourth European Conference on Genetic Programming. 2001.