Representation and Structural biases in CGP

Andrew J. Payne and Susan Stepney
Department of Computer Science, University of York, YO10 5DD, UK

***Abstract*— An evolutionary algorithm automatically discovers suitable solutions to a problem, which may lie anywhere in a large search space of candidate solutions. In the case of Genetic Programming, this means performing an efficient search of all possible computer programs represented as trees. Exploration of the search space appears to be constrained by structural mechanisms that exist in Genetic Programming as a consequence of using trees to represent solutions. As a result, programs with certain structures are more likely to be evolved, and others extremely unlikely.**

We investigate whether the graph representation used in Cartesian Genetic Programming causes an analogous biasing effect, imposing natural limitations on the class of solution structures that are likely to be evolved. Representation bias and structural bias are identified: the rarer "regular" structures appear to be easier to evolve than more common "irregular" ones.

I. Introduction

It has been found that exploration of the search space in GP is constrained by structural mechanisms as a consequence of using trees to represent solutions [5] [6]. This bias means that GP is more likely to evolve certain programs (long skinny trees), and unlikely to evolve others (shorter bushy trees), purely as a result of their structure.

It seems plausible that the class of solutions evolved by other evolutionary algorithms will be similarly limited according to their structure. In general, what structures are easier or harder to evolve?

Cartesian Genetic Programming (CGP) represents programs as indexed graphs. Programs represented as trees in GP are directly manipulated, but genetic operators in CGP act on linear integer genotypes that encode phenotype graphs representing programs. We investigate what impact the graph structure in CGP has on problem difficulty, and what natural limitations it imposes on the class of solutions that are likely to be evolved.

Our evolutionary approach, including a representative set of target graph structures, is described in Section II. Genotypes representing the target graphs are enumerated in Section III to reveal a representation bias. Experimental trials in Section IV reveal a structural bias in terms of the relative evolvability of graph structures. Conclusions are presented in Section V.

II. Evolutionary parameters

The *null hypothesis* for our investigation is that CGP has no structural bias, so all graph shapes require the same effort to evolve. Application specific sets of primitive functions can be assigned to nodes, but phenotype *structure* is relevant to all applications. We consider only the structure of the phenotype, independent from any function.

In tree-based GP a tunably difficult problem based on tree structure has been defined [6] to investigate the structural mechanisms. Trees are described in terms of their depth and their size (number of terminals). The evolvability of trees is investigated as a function of these two parameters. The same approach cannot easily be applied to phenotypes expressed by CGP genotypes because the range of possible graphs cannot be described by just two parameters. Rather than exhaustively search a large parameter space, we examine a small number of specific target graphs with suitably distinct properties.

We use the usual linear genotype representation [13], adapted to our problem of nodes with no assigned function. We assume all nodes have 2 inputs, so each node is represented by 2 genes encoding connectivity. We also assume that the inputs are symmetric (interchangeable). A general CGP program can have multiple program inputs and outputs. For simplicity, we assume only 1 input and 1 output. This allows us to represent the phenotype as a directed acyclic graph (DAG): each active node is represented by a vertex, with additional vertices for the input and output, and each feed-forward connection between nodes is represented by a directed edge. Example genotypes and their phenotypes are shown in figure 1.

A genotype of n nodes and 1 output is represented by $2n+1$ integer-values genes. The genes representing the connectivity of node $1 \leq i \leq n$ are initialised to integer values in the range $[0, i-1]$, while the output gene is initially assigned the value n to connect the output to the final node. When offspring are mutated, each gene is mutated with equal independent probability. The value of a gene always changes as a result of mutation, except for genes encoding the node with index 1, which can only take the value 0. We use a standard $(1+4)$ Evolution Strategy algorithm [14].

Target graph phenotypes with diverse structural properties are selected so that their relative evolvability using CGP can be compared. We select several pairs of targets, with similar structures within each pair, and diverse structures between pairs. These are shown in Figure 1, where they are all illustrated with CGP genotypes of 10 nodes (21 genes), for uniformity. Many other genotypes can express these phenotypes.

The Linear targets **l7** and **l8** are trivially similar. The Tree **t7** and Parity **p9** (based on a textbook parity circuit) also have structural similarity. The smaller Square **s8** is formed by removing one node from the larger Square **s9**. Similarly, the smaller Adder **a7** is formed by removing one node from the larger Adder **a8** (based on the textbook adder comprising 8 NAND gates). A number of CGP genotypes were randomly generated, and those with between 7 and 9 active nodes were collected. Target pairs were formed by identifying phenotypes

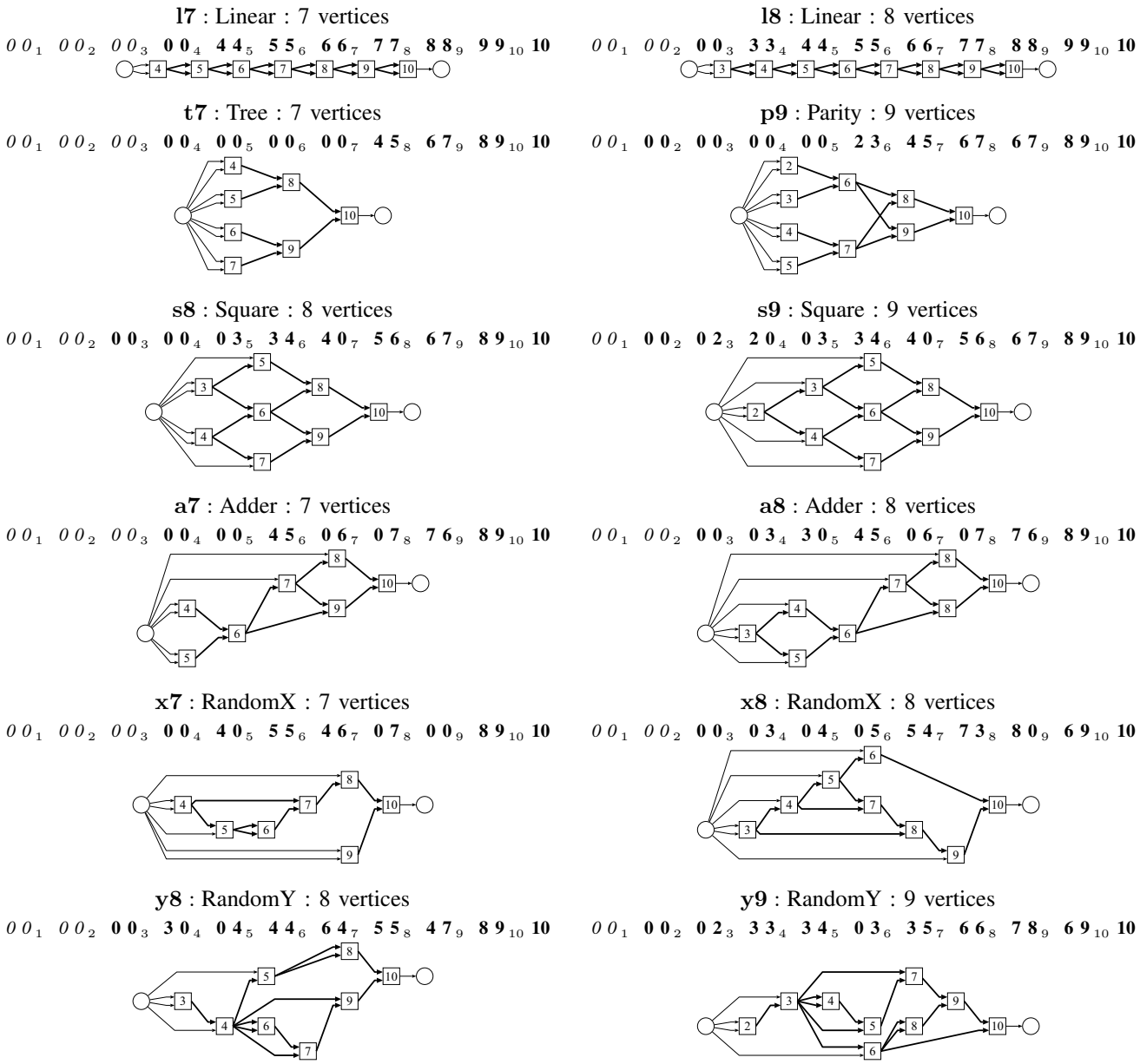

Fig. 1. Genotypes and phenotypes of six target pairs, paired by structural similarity. There is one input (indexed 0), and the nodes in the genotype are indexed from 1 to 10 (added as subscripts for illustration only). The connectivity of each node is represented by a pair of integers. Active (connected) nodes are shown in bold, inactive nodes in italic, for illustration. A final gene encodes which node connects to the output node. Vertices of the phenotype are numbered with the index of the corresponding node in the genotype, again for illustration.

with the smallest non-zero graph edit distances between them. Two such pairs that have no apparent regular structure are used as two additional pairs of targets: **x7** and **x8**, and **y8** and **y9**.

III. Representation Bias

Many different CGP genotypes encode structurally identical phenotypes. There are several reasons for this. The values of genes encoding the input connectivity of an active node can be swapped without affecting the structure of the phenotype because the inputs are considered interchangeable from a structural perspective. Neutral mutation of an inactive gene results in a different genotype, but the phenotype is unaffected. Furthermore, the active nodes need not always appear in the same order or absolute position in the genotype.

The genotype-phenotype mapping means that not all phenotypic structures are equally represented by the range of possible genotypes, even if all genotypes are themselves equally likely to occur. Under-represented phenotypes may prove harder to evolve as a result, with evolution favouring more frequently represented phenotypes. The representation might therefore be a source of bias.

There are two main factors affecting the number of genotypes representing a given phenotype. First is the number of ways the partial order of the DAG nodes can be flattened into

target	active nodes (N)	N-node genotypes	$\log_{10}$(20-node genotypes) fixed output	mutating output
l7	7	1	29.94	31.09
l8	8	1	28.46	29.48
t7	7	80	31.84	33.00
p9	9	320	29.34	30.24
s8	8	1024	31.47	32.49
s9	9	5376	30.56	31.47
a7	7	64	31.75	32.90
a8	8	256	30.87	31.89
x7	7	32	31.45	32.60
x8	8	256	30.87	31.89
y8	8	320	30.96	31.98
y9	9	640	29.64	30.54

Fig. 2. Multiple genotypes can represent the same phenotype. Shown are the number of N-node genotypes (where there are no redundant nodes), and the logarithm of the number of 20-node genotypes (the length used in our experiments), that can represent each target phenotype.

a total order on the genotype. This ranges from 1 for a linear genotype like **l7** and **l8**, to $N!$ for a fully parallel graph of N nodes each connected only to the input and output (allowing all nodes to be connected to the output). This factor depends on the size of the target, and the topology of the target. Second is the number of ways the remaining redundant (unexpressed) genes can be arranged. There is a combinatoric part, as the unexpressed genes are distributed through the expressed nodes, and a further part due to the fact that genes with a higher node number have more states due to the greater number of connection possibilities. This factor depends only on the length of the genotype and the size of the target, not on the topology of the target. The numbers for the selected targets are shown in figure 2.

Target **t7** is the most prevalent: there are 10^{33} mutating output 20-node genotypes that represent this single phenotype. Even so, **t7** is represented by only a small proportion of all possible genotypes, accounting for 1 genotype in 84973 (0.0012% of genotypes) with a fixed output and 1 genotype in 119667 (0.00084% of genotypes) with a mutating output. The vast majority of valid CGP genotypes ($> 99.99\%$) do not represent any of the 12 selected targets. Such a sparse solution density justifies applying an evolutionary algorithm to the task of finding solutions.

The Linear phenotypes are noticeably under-represented. For every genotype representing an **l8** phenotype, there are 1024 genotypes representing an **s8** phenotype, even though both phenotypes have the same number of vertices. This suggests Linear targets may suffer from negative representation bias.

The larger phenotype of a structurally similar pair is expressed by considerably fewer genotypes than its smaller counterpart. This is primarily because there are fewer ways of distributing the active nodes representing a larger graph in the uniform 20-node genotype. Larger phenotypes are likely to be harder to evolve because there are fewer solutions in the search space of all possible genotypes. Yet even the most uncommon genotype (**l8**) is represented by more than 10^{29} mutating output 20-node genotypes.

IV. Structural Bias

The evolutionary algorithm does not sample all genotypes uniformly, but uses a series of point mutations to navigate to positions of optimal fitness in the search landscape. The evolutionary operators define the steps that can be taken through the landscape; the fitness function affects the probability of taking those steps. The chosen genetic operators and evolutionary algorithm may have inherent mechanisms that affect the relative evolvability of graph shapes.

This section investigates the performance of the evolutionary algorithm over CGP structures by performing trials to evolve each target, to establish whether some target phenotype structures are in practice harder to evolve than others. Comparison with the relative number of genotypes representing each target identifies the net effect of any structural bias.

The motivating tree-based GP explorations [6] use a single fitness function. However, the complexity of the fitness landscape (as defined by the fitness function) is often taken as an indicative model of problem difficulty [10]. So here we investigate evolvability of structures using two different fitness functions (graph edit distance, and feature distance), to help distinguish effects due to the target structure rather than to the fitness function itself. The fitness function measures the fitness of the expressed phenotype only, so inactive nodes are not involved in fitness assessment [12].

A. Graph Edit Distance Fitness Function

Our first fitness function is graph edit distance, which describes similarity between graphs by counting the number of edit operations required to transform one graph into another [1]. Edit operations can include insertion, deletion or substitution of an edge or a vertex. Each edit operation has a cost, and the edit distance between two graphs is the sequence of edit operations that transforms one graph to another with minimum cost. This is equivalent to finding the maximum common subgraph [1].

[15] presents a measure of similarity for use with maximum common subgraphs that considers both vertices and edges. This forms the basis for our edit distance cost function:

$$d(g,h) = 1 - \frac{(|V(\mathrm{mcs}(g,h))| + |E(\mathrm{mcs}(g,h))|)^2}{(|V(g)| + |E(g)|)(|V(h)| + |E(h)|)} \qquad (1)$$

where $|V(g)|$ is the number of vertices in graph g and $|E(g)|$ is the number of edges.

Standard maximum common subgraph algorithms are relatively simple to implement, but have high computational complexity, being NP-complete [4]. Hence our investigation of small graphs. We use McGregor's backtracking maximum common subgraph algorithm [11], optimised to take into account the feed-forward nature of the CGP DAGs.

B. Feature Distance Fitness Function

Our second fitness function considers the structural features that two graphs have in common. The structural description of a graph comprising several principal features can be represented as a feature vector. The distance between two

graphs is measured by the Euclidean distance between their respective feature vectors [16]. A feature vector describing a graph should contain sufficient elements to ensure that each target is uniquely described. We use the following 16 feature statistics.

1) Active vertices, $\#v_a$: The number of active nodes.
2) External vertices : The number of active vertices that receive both incoming edges from the input vertex.
3) Internal vertices : The number of active vertices that receive both incoming edges from other active vertices.
4) Double-linked vertices : The number of active vertices that receive both incoming edges from the same source vertex.
5) Number of paths, p : The number of distinct paths from the input vertex to the output vertex.
6) Minimum depth : The number of active vertices in the shortest path from the input vertex to the output vertex.
7) Mean depth : The mean number of active vertices in all paths from the input vertex to the output vertex.
8) Maximum depth : The number of active vertices in the longest path from the input vertex to the output vertex.
9) Graph width : The number of active vertices divided by the mean depth. (This derived statistic is included because it describes an important structural property.)
10) External path length : The sum of the lengths of all distinct paths from the input vertex to the output vertex.
11) Internal path length : The sum of the lengths of all distinct paths from each active vertex to the output vertex.
12) Visitation length : The sum of the internal path length, external path length and number of active vertices. (Originally proposed for trees [7], it is included here for completeness.)
13) Vertex depth : The total depth d_t of an active vertex is the sum of its depth over every path through the graph. The vertex depth is $\sqrt{\sum_{v_a} d_t^2}/p$.
14) Vertex reuse : The reuse r of an active vertex is the number of paths from that active vertex to the output vertex. The vertex reuse is $\sqrt{\sum_{v_a} r^2}/p$.
15) Vertex fan-in : The fan-in f_i of an active vertex is the number of paths from the input vertex that enter that active vertex. The vertex fan-in is $\sqrt{\sum_{v_a} f_i^2}/p$.
16) Vertex out-degree : The out-degree f_o of an active vertex is the number of directed edges leaving that vertex. The vertex out-degree is $\sum_{v_a} f_o^2/\#v_a$.

There is correlation between several of the chosen feature statistics, but each possible phenotypic structure should ideally be described by a unique set of statistics. The likelihood of this is increased by using a greater number of feature statistics that describe a graph's structural properties in different ways. Derived statistics like graph width and visitation length are included because they describe the graph in meaningful terms in their own right.

This set of feature statistics to describe a phenotype graph can be calculated in $O(n)$ time, where n is the number of nodes in the CGP DAG genotype. A fitness function based on feature statistics is therefore much more computationally efficient than calculating graph edit distance using the maximum common subgraph.

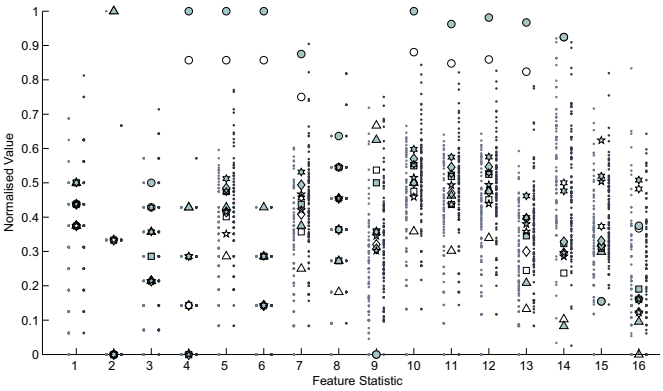

Fig. 3. Distribution of normalised feature statistics, for all targets (symbols: see figure 11 for key), and randomly sampled genotypes (dots). For each statistic, the left column of dots are random genotypes with mutating output, the right-hand column are random with fixed output.

Some feature statistics vary over orders of magnitude, which means they would dominate the calculated feature distance. This applies to the number of paths through the graph, the external and internal path lengths, and the visitation length. We reduce these ranges by taking logs (base 2) of these statistics.

Finally, we normalise the feature statistics so that all components make a similar contribution to the computed distance. Some feature statistics have obvious bounds, such as the number of active vertices expressed by a 20-node genotype, which always occupies the range $[1, 20]$. Others do not have an easily calculated upper bound, which limits scope for perfect normalisation. Instead, we find minimum and maximum values of each feature statistic $stat_i$ of all targets, and of the graphs expressed by 1000 randomly sampled genotypes, finding min_i and max_i. We apply a linear normalisation process to each statistic $stat_i$ according to equation (2), yielding normalised statistics filling the range $[0, 1]$ (with the possibility that some evolved graphs may have values outside this range).

$$\mathrm{norm}(stat_i) = \frac{stat_i - min_i}{max_i} \tag{2}$$

Figure 3 shows how the normalised feature statistics for each target, and graphs expressed by randomly sampled genotypes, are distributed.

The *normalised feature distance* between two graphs is the Euclidean distance between their respective feature vectors, having taken $\log_2$ of the identified feature statistics and normalised all statistics to fit the range $[0, 1]$ as described above.

C. Experimental Procedure

The relative evolvability of the targets is found experimentally by performing a series of trials. Each trial consists of a number of independent runs of the CGP algorithm, aiming to evolve one particular target.

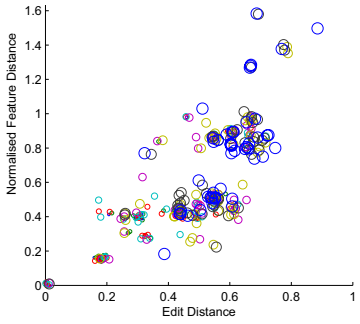

Fig. 4. Correlation between fitness functions. Starting from target **t7**, a series of 1–8 random point mutations is made, and the two fitness functions calculated. (Larger circles correspond to longer series of mutations.) Spearman's non-parametric rank correlation coefficient [8] $\rho = 0.862$; all other starting targets have lower correlation coefficients.

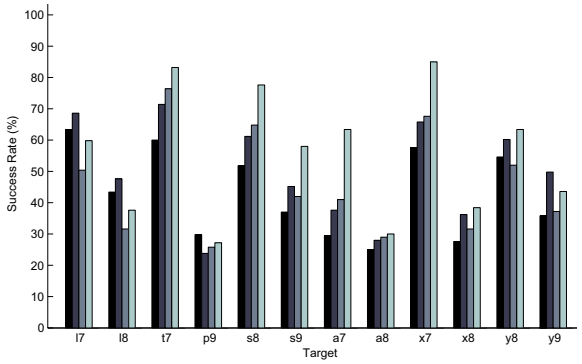

Fig. 5. Success rate in trials of 500 runs using edit distance. Bars for each target represent Fixed output 5% mutation, Fixed 10%, Mutating 5%, and Mutating 10%. (The same convention is used in the following graphs.)

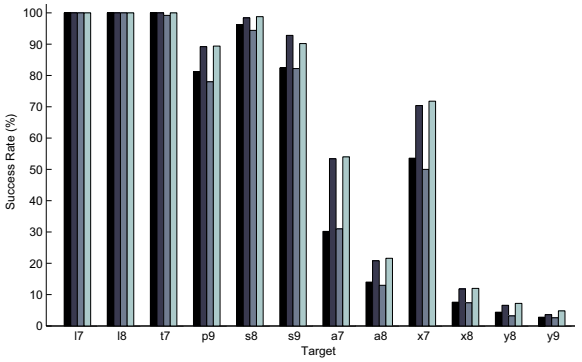

Fig. 6. Success rate in trials of 500 runs using normalised feature distance.

A uniform genotype of 20 nodes (41 genes, including the output) is used in all trials. The genetic operator is point mutation. The following parameters are varied independently.

Mutation: Separate trials are carried out in which each gene has an independent 5% or 10% probability of mutation. In a genotype of 20 nodes (40 genes, not including the output) this means an average of 2 or 4 genes are mutated in each genotype offspring.

Output: Separate trials are carried out in which the output is permanently connected to the final node, and ones in which the output gene is mutated with the same probability as any other gene.

Fitness function: Separate trials are carried out for each fitness function. Because we have not proved that our feature distance fitness function returns 0 only when the phenotype graphs are identical, each such candidate solution is also checked using the graph edit distance. Figure 4 demonstrates that, although there is a correlation between the two fitness functions, it is not strong.

This gives $2 \times 2 \times 2 = 8$ trials per target.

Preliminary trials of 20 runs of up to 5000 generations were performed for each target. 58% of the runs achieved success within 5000 generations, but 53% of runs had achieved success within just 2000 generations, accounting for over 90% of the successful runs. In those runs that failed to achieve success within 2000 generations, the best fitness improved rapidly at the start of a run before reaching a plateau of no further improvement. In the main experiments, each trial consists of 500 runs, each of up to 2000 generations.

D. Results

The success rates achieved in each trial are shown in Figures 5 and 6. It is immediately clear that not all targets are equally evolvable.

Mutating genes with the higher 10% probability mostly improves the success rate. Using genotypes with a mutating output improves the success rate for most targets when using edit distance, but has essentially no effect when using feature distance. The biggest effect on success rate is the fitness function, with the feature distance greatly improving the success rate for the most "regular" structures, and greatly reducing it for the most "random" structures.

Relative evolvability is also measured by the computational effort, which indicates the minimum number of genotypes that would need to be processed in order to find a solution with 99% probability [9]. Computational effort finds the optimal length of an evolutionary run, the number of such runs required to achieve a 99% success rate and hence the number of genotypes that would be processed during those runs. (The Computational Effort statistic has been subject to criticism, for example [2], since its biases make it difficult to compare across different experiments. However, here we are using it to compare results obtained under the same experimental conditions.) The computational effort calculated for each trial is shown in Figures 7 and 8.

E. Comparison with Random Sampling

The experimental success rate can be compared with the probability of finding a solution by random sampling of genotypes. This compensates for the effect of *representation* bias: we expect search to be better than random sampling; if there is no further *structural* bias, we would expect the improvement to be independent of target structure.

The probability p_g that any one randomly sampled genotype represents a particular target is easily calculated from the number of 20-node genotypes that represent that target, shown

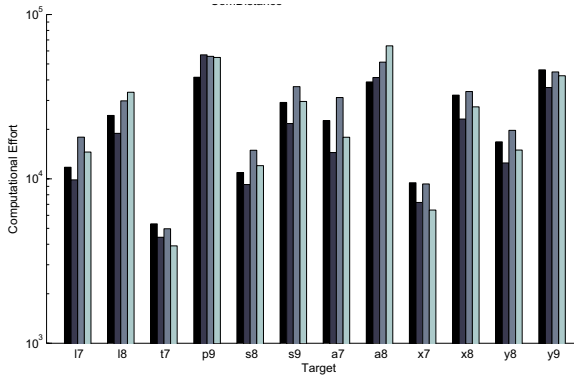

Fig. 7. Computational effort in trials of 500 runs using edit distance. (Note the log scale.)

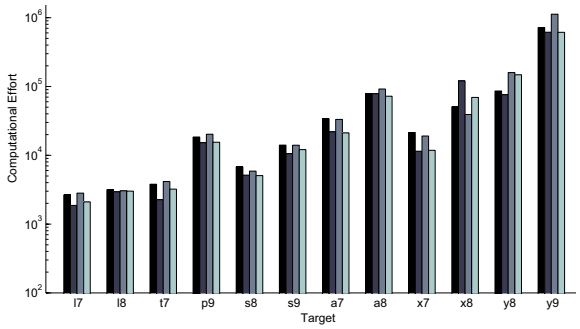

Fig. 8. Computational effort in trials of 500 runs using normalised feature distance.

in figure 2. Running the CGP algorithm for up to 2000 generations using $(1 + 4)$ Evolution Strategy involves up to 8001 genotype evaluations. The probability p_s of finding a solution by randomly sampling 8001 genotypes is given by:

$$p_s = 1 - (1 - p_g)^{8001} \tag{3}$$

A similar approach is adopted to evaluate the effectiveness of CGP on a Boolean parity problem [3].

The minimum computational effort required to find a solution by random sampling of genotypes can be calculated in a similar manner. Computational effort calculations require the cumulative probability of success $P(M, k)$ attained by generation k with population size M. The probability of success by random sampling of genotypes at the equivalent of generation k is:

$$P(M, k) = 1 - (1 - p_g)^{kM+1} \tag{4}$$

where $M = 4$ for CGP using $(1 + 4)$ Evolution Strategy. The standard calculation of minimum computational effort can be used once the values of $P(M, k)$ have been found.

Figures 9 and 10 show the improvement in success rate and computational effort using CGP over the calculated figures using random genotype sampling. Unsurprisingly, CGP consistently out-performs random sampling of genotypes for all targets. However, the degree of improvement varies markedly (from many orders of magnitude, to a factor of a few), indicating a strong structural bias.

	success rate improvement		*computational effort reduction*	
Target	5%	10%	5%	10%
l7	537	581	0.04%	0.03%
l8	10850	11900	0.00%	0.00%
t7	7	8	1.36%	1.13%
p9	993	793	0.03%	0.05%
s8	13	16	1.18%	1.00%
s9	75	91	0.39%	0.29%
a7	4	5	4.62%	2.95%
a8	25	28	1.05%	1.12%
x7	16	18	0.97%	0.73%
x8	28	37	0.87%	0.62%
y8	44	49	0.57%	0.42%
y9	607	844	0.07%	0.06%

Fig. 9. Success rate improvement (success rate of CGP / success rate of random sampling) and computational effort reduction (CE of CGP / CE of random sampling), for 5% and 10% mutation, fixed output, graph edit distance.

	success rate improvement		*computational effort reduction*	
Target	5%	10%	5%	10%
l7	847	847	0.01%	0.01%
l8	25000	25000	0.00%	0.00%
t7	11	11	0.97%	0.58%
p9	2707	2973	0.01%	0.01%
s8	25	25	0.73%	0.56%
s9	166	187	0.19%	0.14%
a7	4	7	6.95%	4.48%
a8	14	21	2.10%	2.12%
x7	15	19	2.19%	1.17%
x8	8	12	1.37%	3.26%
y8	4	5	2.90%	2.56%
y9	47	61	1.15%	0.99%

Fig. 10. Success rate improvement (success rate of CGP / success rate of random sampling) and computational effort reduction (CE of CGP / CE of random sampling), for 5% and 10% mutation, fixed output, feature distance.

The correlation between the computational effort for the edit distance and random sampling cases is shown in Figure 11, to help expose correlations. Vertical separation between a homogeneous pair of data points at the same horizontal position represents the difference between mutating genes with a 5% and 10% probability. The effect this has on evolvability is generally insignificant compared with the relative difficulty of evolving different targets.

The Linear targets (**l7** and **l8**, represented by circle markers) are the greatest outliers. Their position indicates that CGP is able to evolve these targets relatively more easily than one might expect from their solution density in all valid genotypes. Conversely, the position of the Adder targets (**a7** and **a8**, represented by diamond markers) indicates that they present a greater challenge to CGP than might be expected. These observations also hold in the feature distance case.

F. Observations on Edit Distance

The Tree target (**t7**) requires the least computational effort. This is closely followed by the smallest Random target (**x7**), which is evolved with a comparable rate of success. A much lower success rate is achieved with the larger Adder target (**a8**) and the Parity target (**p9**), which both require considerably more effort.

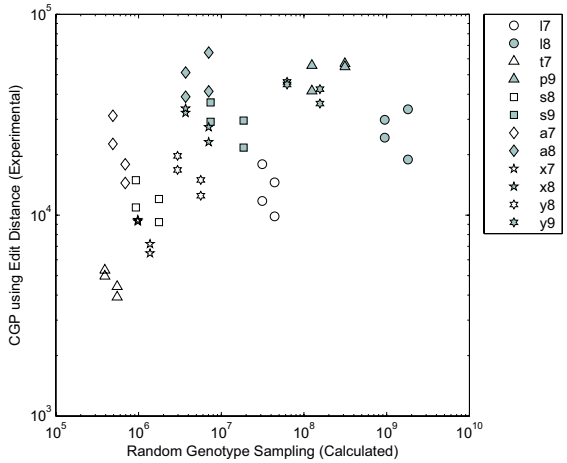

Fig. 11. Correlation between computational effort for edit distance and random sampling cases. Targets are identified by the marker style: the shape identifies the pair of structurally similar targets; shaded markers represent the larger target within the pair. A data point is included for each combination of 5% and 10% mutation probability with a fixed and mutating output, so there are 4 data points for each target.

A series of Wilcoxon rank sum tests [17] compares the distribution of generations required to evolve the **t7** target against the generations required to evolve all other targets. This yields p-values that are well below 0.01, revealing a statistically significant difference at the 99% significance level. A similar series of tests for the **x7** target also reveals a statistically significant difference at the 99% significance level, in trials where the output gene can be mutated.

The smallest Random target (**x7**) appears to be one of the easier targets to evolve, whereas targets **x8** and **y9** are among the hardest. The experimental trials reveal no separation in terms of evolvability between the targets derived from randomly generated genotypes and those with more regular structures. This provides no evidence to suggest that irregular structures are harder to evolve than regular structures on account of their randomness.

Visual inspection of the experimental results suggests that the larger target of a structurally similar pair requires more computational effort than its smaller counterpart. Further Wilcoxon rank sum tests compare the distributions of generations required to evolve the larger and smaller targets. A statistically significant difference at the 99% significance level is found between the larger and smaller target of all pairs except for the Adders (**a7** and **a8**) when a 5% mutation rate and a fixed output is used.

The Linear targets (**l7** and **l8**) are evolved relatively more easily than one might expect from the representational bias, while the Adder targets (**a7** and **a8**) exceed their expected difficulty. This indicates a bias in the evolutionary process that does not merely follow the representation bias imposed by the genotype-phenotype mapping. The fact that one set of outliers represents both Linear targets, and the other both Adder targets, suggests phenotypic structure is a likely factor.

G. Observations on Feature Distance

The difference in evolvability between targets is more pronounced when using feature distance than when using edit distance. A 100% success rate is achieved when evolving the Linear targets **l7** and **l8** using feature distance. In fact, the smaller Linear target **l7** is always evolved within 1000 generations. Significantly lower success rates are achieved with the larger Random targets. Indeed, the Random **y9** target is not evolved in any of the 500 runs using a fixed output and a 10% mutation rate.

There is greater consistency in success rate and required computational effort between pairs of structurally similar targets, although the larger target of a pair is still the harder to evolve. In particular, the paired Tree target **t7** and Parity target **p9** achieve much more similar success rates than those achieved using graph edit distance. Wilcoxon rank sum tests [17] compare the distributions of generations required to evolve the larger and smaller targets in each pair. The generations required to evolve paired targets have significantly different distributions at the 99% significance level, but larger p-values generally occur between paired targets than between non-paired targets, indicating that structurally similar paired targets are more similarly evolvable. The largest p-values occur between the Linear targets **l7** and **l8** and between the Square targets **s8** and **s9**. There is greater distinction between the evolvability of paired targets that have less regular structures.

The Linear targets **l7** and **l8** benefit the most from the change in fitness function, but evolvability of the Tree target **t7** is almost unchanged. It is somewhat surprising that the Tree target is no longer easier to evolve than the Square targets **s8** and **s9** despite being smaller, more frequently occurring in randomly sampled genotypes, and having more extreme feature statistics (figure 3). Structural similarity has a greater effect on evolvability under feature distance than the size of the target or the frequency with which it is represented by randomly sampled genotypes.

Evolving targets with more irregular structures requires greater effort with the feature difference fitness. This can be explained by the distributions of feature statistics, as the statistics describing many of the targets are relatively tightly clustered in areas also occupied by statistics describing the graphs expressed by randomly sampled genotypes (figure 3). The normalised feature distance imposes a structural bias in favour of regular structures described by extreme feature statistics. Phenotype size is also a factor in the relative evolvability of irregular targets, as evolving the larger target of a structurally similar pair requires more effort.

H. Fitness Function Dominance

The chosen fitness function appears to be responsible for many of the observed differences between separate experimental trials. (Note that [5] [6] used a single fitness function in their trials.) One observation that is common across all trials is that less computational effort is required to evolve the smaller target of a structurally similar pair than its larger counterpart. This is only partly explained by the relative numbers of

genotypes representing smaller and larger targets, resulting from the genotype-phenotype mapping (figure 2). In other respects, the chosen fitness function has an influence on the relative evolvability of targets that may mask any underlying mechanisms of CGP.

Evolvability using CGP is partly influenced by the distribution of genotypes representing a particular target, but this is not the only factor. The Linear targets **l7** and **l8** are under-represented in the genotype distribution, but are among the easiest targets to evolve. This might be partly explained by the range of likely structural changes resulting from CGP point mutations, analogous to structural mechanisms in GP. However, the chosen fitness function has a significant effect on the relative evolvability of targets.

V. Conclusions

Some phenotypes are significantly more difficult to evolve than others, based purely on their structure. This has been demonstrated only under the particular algorithm parameters used in the experimental trials. Parameters such as mutation probability and whether the output gene can be mutated were varied, generally without a statistically significant effect. The fitness function had a significant effect. Other parameters such as the choice of $(1+4)$ Evolution Strategy remained fixed, as this is representative of many existing CGP implementations.

1) Not all structures are equally evolvable: Lower success rates are achieved, and greater computational effort is required, when evolving certain target shapes. This demonstrates that some phenotypes are more difficult to evolve than others, purely as a result of their *structure*, not of their *function*. This has implications for all applications of CGP, because it can not be assumed that all valid solutions to a problem are equally likely to be found.

2) Genotype-phenotype mapping imposes a representation bias: A bias occurs as a result of using a fixed-length linear genotype to represent phenotype graph structures with limited feed-forward connectivity. Genotype length imposes a limit on the represented phenotype size. A larger phenotype is represented by considerably fewer genotypes than a smaller structurally similar phenotype. As a result, evolvability of structurally similar phenotypes gets harder as phenotype size increases.

Phenotype size is not the only factor affecting the proportion of genotypes representing a particular phenotype. Feed-forward connectivity restricts the order in which active nodes may appear in genotypes representing a phenotype structure. CGP consistently out-performs random sampling of genotypes, but there is (a weak) correlation between the proportion of genotypes representing a particular phenotype and the computational effort required to evolve it.

3) Structural mechanisms impose an additional structural bias: There is not always direct correlation between evolvability and the distribution of genotypes. Phenotypes with a simple linear structure are under-represented in the genotype distribution, but are found to be among the easiest to evolve. This might be partly explained by the range of likely structural changes resulting from CGP point mutations, analogous to structural mechanisms in GP.

Experimental trials reveal that simple linear and tree structures benefit most from this bias. These structures share the properties of self-similarity and simple connectivity, and can be extended incrementally by a series of point mutations. More complex or dense structures with greater degrees of interconnectivity prove a greater challenge, perhaps because many vertex connections are involved in the process of inserting a subgraph. Further analysis is required to characterise the structures that benefit and suffer in terms of evolvability.

Representation and structural biases have been uncovered. The fact that biases exist is not surprising: what has been demonstrated here is the size and direction of these biases. In particular, the representation bias is outweighed by the structural bias, and the rarer "regular" structures appear to be easier to evolve than the more common and seemingly more "irregular" ones. A bias towards more regular solutions might be viewed as a good thing, unless your particular problem requires an irregularly-structured solution.

References

[1] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[2] S. Christensen and F. Oppacher. An analysis of Koza's computational effort statistic for Genetic Programming. In *EuroGP 2002*, number 2278 in LNCS, pages 182–191. Springer, 2002.

[3] M. Collins. Finding needles in haystacks is harder with neutrality. In *GECCO 2005*, pages 1613–1618. ACM Press, 2005.

[4] D. Conte, P. Foggia, and M. Vento. Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs. *Journal of Graph Algorithms and Applications*, 11(1):99–143, 2007.

[5] J. M. Daida and A. M. Hilss. Identifying structural mechanisms in standard Genetic Programming. In *GECCO 2003*, number 2724 in LNCS, pages 1639–1651. Springer, 2003.

[6] J. M. Daida, H. Li, R. Tang, and A. M. Hilss. What makes a problem GP-hard? Validating a hypothesis of structural causes. In *GECCO 2003*, number 2724 in LNCS, pages 1665–1677. Springer, 2003.

[7] M. Keijzer and J. Foster. Crossover bias in Genetic Programming. In *EuroGP 2007*, number 4445 in LNCS, pages 33–43. Springer, 2007.

[8] M. Kendall and J. D. Gibbons. *Rank Correlation Methods*. Edward Arnold, 1990.

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

[10] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, 2002.

[11] J. J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*, 12:23–34, 1982.

[12] J. F. Miller. What bloat? Cartesian Genetic Programming on Boolean problems. In *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 295–302, 2001.

[13] J. F. Miller and S. L. Smith. Redundancy and computational efficiency in Cartesian Genetic Programming. *IEEE Transactions on Evolutionary Computation*, 10(2):167–174, 2006.

[14] J. F. Miller and P. Thomson. Cartesian Genetic Programming. In *EuroGP 2000*, number 1802 in LNCS, pages 121–132. Springer, 2000.

[15] J. W. Raymond, E. J. Gardiner, and P. Willett. RASCAL: calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631–644, 2002.

[16] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3):353–362, 1983.

[17] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945.