Noname manuscript No.
(will be inserted by the editor)

Neutral Genetic Drift: An Investigation using Cartesian Genetic Programming

Andrew James Turner · Julian Francis Miller

the date of receipt and acceptance should be inserted later

Abstract Neutral genetic drift is an evolutionary mechanism which can strongly aid the escape from local optima. This makes neutral genetic drift an increasingly important property of Evolutionary Computational methods as more challenging application are approached. Cartesian Genetic Programming is a Genetic Programming technique which contains explicit, as well as the more common implicit, genetic redundancy. As explicit genetic redundancy is easily identified and manipulated it represents a useful tool for investigating neutral genetic drift.

The contributions of this paper are as follows. Firstly the paper presents a substantial evaluation of the role and benefits of neutral genetic drift in Cartesian Genetic Programming. Here it is shown that the benefits of explicit genetic redundancy are *additive* to the benefits of implicit genetic redundancy. This is significant as it indicates that that levels of implicit genetic redundancy present in other Evolutionary Computational methods may be insufficient to fully utilise neutral genetic drift. It is also shown than the identification and manipulation of explicit genetic redundancy is far easier than for implicit genetic redundancy. This is significant as it makes the investigations here possible and leads to new possibilities for allowing more effective use of neutral genetic drift. This is the case not only for Cartesian Genetic Programming, but many other Evolutionary Computational methods which contain explicit genetic redundancy. Finally, it is also shown that neutral genetic drift has additional benefits other than aiding the escape from local optima.

Andrew James Turner
The University of York
Electronics Department
Intelligent Systems Group
York, UK
E-mail: andrew.turner@york.ac.uk

Julian Francis Miller
The University of York
Electronics Department
Intelligent Systems Group
York, UK
E-mail: julian.miller@york.ac.uk

1 Introduction

Sewall Wright first introduced the notion that natural evolution is searching over a space of possible solutions [50]. This space of solutions, commonly referred to as a fitness landscape, gives the fitness of each possible solution in relation to one another. This concept was later adopted by the machine learning literature, particularly in Evolutionary Computation (EC) [23].

One of the insights gained from visualising the fitness landscape is the idea of local optima. These are sub-optimal areas of the search space where most, if not all possible mutations result in a worse fitness. As a result, escaping local optima is challenging. Typically mutation is thought of as the mechanism for escaping local optima, but mutation alone is often insufficient depending upon the local fitness landscape. Another mechanism for escaping local optima is provided by the presence of neutral genetic drift (NGD). NGD was also first proposed in the domain of evolutionary biology by Motoo Kimura [19]. NGD is where mutations are made to a genotype which do not influence its semantics but are still selected for the next generation. This causes the position within the fitness landscape to change without requiring fitness improving mutations. Once in a new area of the fitness landscape the possible solutions accessible though mutation are different to what was possible before the neutral mutation(s). Therefore when in local optima the presence of NGD causes the number of possible solutions one mutation away to *vary*, thus aiding the escape from local optima.

An additional consequence of NGD is that it can also occur when the search is not trapped in local optima. This can happen if neutral mutations take place at the same time as fitness improving mutations (i.e. the inactive mutations 'hitch-hike'). Although in this scenario NGD cannot be aiding the escape from local optima, it may still be influencing the evolutionary search.

It is important to understand and utilise the role of NGD in Genetic Programming (GP) [22,33], and EC in general, as it represents a powerful mechanism for searching the fitness landscape. Additionally, as more challenging applications are investigated the benefits of NGD become more significant. This is because more challenging applications are likely to represent fitness landscapes with more numerous local optima. As NGD is thought to aid the escape from local optima it becomes increasingly significant as the difficulty of the landscape increases.

In tree-based GP it is common to refer to inactive/redundant genes[1] as being those which are present in the genotype but which do not affect the semantics. An example of this form of genetic redundancy is where a section of the phenotype is multiplied by zero and hence does not contribute to the programs output; see Figure 1. In this paper, this type of redundancy is referred to as *implicit genetic redundancy* as the genes are decoded into the phenotype, so in that sense are active, however they do not contribute to the semantics of the phenotype. Here NGD of implicitly redundant genetic material is referred to Implicit Neutral Genetic Drift (INGD) to distinguish it from NGD in general and other forms of NGD.

Interestingly, many other forms of GP contain a more explicit form of genetic redundancy. Cartesian Genetic Programming (CGP) [26], a graph-based form of GP, and Linear Genetic Programming (LGP) [4], a register based form of GP, con-

[1] The terms inactive genes and redundant genes are used interchangeably and refer to genes which do not influence the semantics and/or the phenotype.

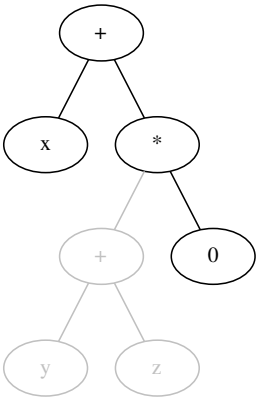


Fig. 1: Implicit genetic redundancy in tree-based GP. Active nodes are shown in black, implicitly redundant nodes in grey.

tain inactive genes which are not decoded into the phenotype. Push-GP is a stack based representation in which evolved programs manipulate program and data via stacks [37]. Push-GP also exhibits explicit genetic redundancy since instructions can be left on the CODE or EXEC stacks which are ignored. In the case of Grammatical Evolution, genetic redundancy can be introduced in the underlying binary representation. An extensive study of this is given in [49]. Both CGP and LGP employ a genotype-phenotype mapping which removes explicitly redundant genes [53] or structurally noneffective operations [4]. This form of genetic redundancy is referred to as *explicit genetic redundancy*. Again as before NGD acting on explicitly redundant genetic material is referred to as Explicit Neutral Genetic Drift (ENGD). Forms of GP other than tree-based GP also contains implicit genetic redundancy via the same mechanism described for tree-based GP and so therefore also exhibit INGD.

One of the difficulties in studying NGD is identifying which genes are genetically redundant. For instance identifying implicitly redundant genes in tree-based GP is challenging [3] as it is involves analysing the semantics of the program. However, identifying explicitly inactive genes in CGP is trivial as they are the genes not associated with the nodes connecting inputs to outputs[2]. This makes CGP a useful tool for investigating the role of NGD. As is shown in this paper, by using a range of restrictions to CGP various aspects of INGD *and* ENGD can be isolated and studied independently.

Previous research investigating NGD in CGP [26,46] examined the effect of removing NGD by preventing the selection of child chromosomes over their parents

[2] Identifying which genes are active is recommended in [26] and normally included in most CGP implementations (e.g. [39]). Doing so considerably saves computation time by avoiding calculating the outputs of unused nodes.

if they have equal fitness. This methodology prevents both INGD and ENGD aiding the escape from local optima. The experiments undertaken did not isolate the benefit of ENGD; the type of NGD present in CGP but not in tree-based GP. Additionally the methodology did not consider potential benefits of NGD other than escaping local optima; such as NGD occurring at the same time as fitness improving mutations to active genes.

Other investigations of genetic redundancy in CGP looked at the effect of increasing the levels of explicit genetic redundancy [27]. This was achieved by increasing the number of available nodes which increased the proportion of inactive genes. The work reported that increasing the level of explicit genetic redundancy consistently improved the evolutionary search. However it is clear that this tread cannot continue indefinitely since it implies that using an infinite number of nodes would solve a given task in zero time. An alternative explanation, other than the increase in explicit genetic redundancy, has also since been presented. It speculates that the reason using high numbers of available nodes is beneficial is because it compensates for the length bias in CGP [14, 15].

The contribution of this paper is a substantial evaluation of the role of NGD in CGP. The benefits of both INGD and ENGD are rigorously evaluated including aspects other than aiding the escape from local optima. Additionally, past experiments are repeated and extended in order to confirm previous results whilst offering further insights. The paper demonstrates that both INGD and ENGD are highly beneficial to the evolutionary search of CGP and their role in allowing escape from local optima is not the sole reason for the improvements identified. Additionally, the previous misconception that increasing the level of explicit redundancy, by increasing the number of nodes, always improves the evolutionary search is refuted and indeed it is shown that there are thresholds in genotype size above which the performance of evolutionary search deteriorates.

Although the focus of this paper is NGD in CGP, the results presented are relevant to other GP and EC techniques. For instance, it is empirically demonstrated that NGD provides benefits other than merely aiding the escape from local optima; a benefit likely present in many EC techniques which also contain genetic redundancy. Additionally it is shown that ENGD is beneficial and additive to INGD. Therefore the inclusion of ENGD in current or future EC methods is likely to be beneficial.

The remainder of this paper is as follows. Section 2 introduces CGP. Section 3 discusses why genetic redundancy is thought to offer an advantage with a focus on CGP. Section 4 discusses the benchmarks which are used in the presented investigations. Section 5 then describes and presents the investigations into NGD in CGP followed by Section 6 presenting the investigations into varying the levels of explicit genetic redundancy in CGP. Then Sections 7 and 8 respectively give a discussion of the results and conclusions.

2 Cartesian Genetic Programming

CGP [26,28] is a form of GP [22,33] which typically evolves acyclic computational structures of nodes (graphs) indexed by their Cartesian coordinates. However, it should be noted that CGP has also recently been extended to be capable of creating recurrent (cyclic) computational structures [40–42]. It is well-known that CGP does not suffer from bloat [25,38]; a handicap of many GP methods [36]. CGP chromosomes contain explicitly inactive or non-functioning genes which enable ENGD during evolution [46,51]. CGP typically uses point or probabilistic mutation, no crossover [5] and a $(1 + \lambda)$-ES. Although CGP genotypes are of static size, the number of active genes can vary during evolution thus allowing the length of the phenotype to vary. In CGP, the *maximum* number of available nodes is specified in advance, however, in practice only a proportion will be active. Overestimating the number of available nodes has been shown to greatly aid evolution [27].

Each CGP chromosome comprises function genes (F_i), connection genes ($C_{i,j}$) and output genes (O_i); where i indexes each node and j indexes each node's inputs. The function genes represent indexes in a function look-up-table and describe the functionality of each node. The connection genes determine where each node acquires its inputs. For standard CGP (acyclic), connection genes may connect a given node to any previous node in the program, or any of the program inputs. The output genes address any program input or internal node and define which nodes are used to provide the program outputs.

Originally CGP programs were organized with nodes arranged in rows (nodes per layer) and columns (layers); with each node indexed by its row and a column. However, this is an unnecessary constraint as any configuration possible using a given number of rows and columns is also possible using one row with many columns; provided the total number of nodes remains constant. This is because CGP can evolve where each node obtains its inputs. Consequently, in this paper the chromosomes are chosen with one row and n columns; with each node only indexed by its column. A generic (one row) CGP chromosome is given in Equation 1; where α is the arity of each node, n is the number of nodes and m is the number of program outputs.

$$F_0C_{0,0}...C_{0,\alpha-1}...F_{n-1}C_{n-1,0}...C_{n-1,\alpha-1}\colon O_0...O_{m-1} \tag{1}$$

An example CGP program is given in Figure 2 along with its corresponding chromosome; for clarity and following previous conventions the function genes are underlined. As can be seen, all nodes are connected to previous nodes or program inputs. Not all program inputs have to be used, enabling evolution to decide which inputs are significant. An advantage of CGP over tree-based GP, again seen in Figure 2, is that node outputs can be reused multiple times, rather than requiring the same value to be recalculated if it is needed again. Another advantage is that CGP can easily be applied to multiple-input multiple-output problems. Finally, not all nodes contribute to the final program output, these represent the inactive nodes (explicitly redundant genes) which enable ENGD and in turn make variable length phenotypes possible.

CGP chromosomes are initialised by selecting random valid alleles for each gene in a fixed length genotype. As not all genes are decoded into the phenotype this enables an initial population of variable length solutions and explicitly inactive

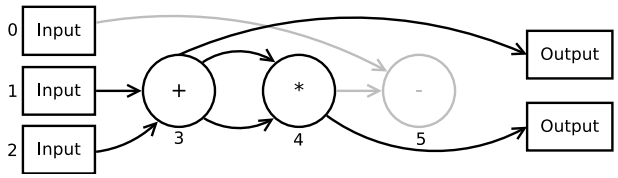


Fig. 2: Example CGP program with three inputs, three available nodes and two outputs. The active genes are shown in bold with the inactive in grey. The chromosome corresponding to the given program is as follows: 012 233 104 3 4

genes. The size of the phenotypes can then be adapted by making more or less of the genes active. Conversely tree-based GP uses variable length chromosomes. These are typically initialised as small genotypes which are directly mapped to small phenotypes. The length of the phenotype is then altered by the addition or removal of genes from the genotype. In this regard CGP and tree-based GP represent two different methods for allowing variable length solutions.

3 Redundancy in Cartesian Genetic programming

The role of redundancy in EC and its impact on neutrality, has been widely studied: [1, 2, 7–12, 17, 18, 34, 49] and recently reviewed [13]. There are many forms of redundancy in GP. For instance, as has been previously discussed, genes can be explicitly or implicitly redundant. It is possible for many genotypes to produce the same phenotypes; this creates redundancy in the genotype-phenotype mapping. It is also possible for many genotypes to produce phenotypes with the same semantics. This creates redundancy in the genotype-semantic mapping. Additionally the solution spaces themselves can contain redundancy with many possible solutions being awarded equal fitness. Finally redundancy can occur in the imprecision of implementing abstract computer programs on real world hardware. For instance the imprecision of floating point operations causing two differing solutions to be awarded the same fitness.

CGP chromosomes contain genes which are ignored in the construction of the phenotype. These inactive genes are therefore genetically redundant. However, such redundant genes can later become active via mutation. For instance a mutation could cause an active node to connect to a previously inactive node, thus making the inactive node active. This type of genetic redundancy is referred to as explicit genetic redundancy because the genes are explicitly removed during the genotype-phenotype decoding stage. As discussed, implicit genetic redundancy is also possible in CGP where a section of the *phenotype* has no influence on it's semantics.

It is worth noting that it has been previously claimed that the presence of explicit genetic redundancy in CGP helps the evolutionary search on "needle in the the haystack" type problems [52]. However, this was refuted and an alternative explanation given in [6] where it was concluded that explicit genetic redundancy does not aid the evolutionary search on needle-in-haystack type problems. However this conclusion said nothing about the merits or demerits of explicit genetic redundancy in CGP generally i.e. when applied to non needle-in-haystack prob-

lems. That is to say explicit genetic redundancy could be, and is indeed shown here to be, generally beneficial.

The remainder of this section discusses NGD with a focus on CGP followed by discussion of previous work which has attempted to increase the levels of genetic redundancy in CGP in order to improve the evolutionary search. For a further discussion of genetic redundancy and NGD in CGP see [26, 46, 51, 53].

3.1 Neutral Genetic Drift

The idea that NGD might be beneficial was first proposed in the field of biological evolution in a highly influential paper *Neutral Theory of Molecular Evolution* [19]. The arguments given apply to NGD caused by both implicit and explicit genetic redundancy.

NGD describes the change in inactive genetic material during evolution. The *neutral* refers to it having no effect on the semantics of the phenotype (fitness), the *genetic* refers to it acting upon the genetic material and the *drift* refers to the genetic material drifting i.e. it is changing in an unguided manner. One of the reasons NGD is thought to be important is because it can lead to genetically (not phenotypically) diverse populations even when trapped in local optima.

Consider an evolutionary run which has reached a local optimum. Over time the population will converge on the best found solution. This is because, when trapped in a local optimum, most mutations will create worse children than their parents and therefore only children which are genetically similar to their parents are selected and survive. This causes the entire population to become genetically similar. If all members of the population are very similar, then the number of possible solutions which can be reached via mutation (or crossover) in one generation is vastly reduced. Thus escaping the local optimum becomes increasingly difficult.

When redundant genes are present, the active genes will still converge but there is no pressure for the inactive genes to converge; as they do not affect fitness and are not guided by selection. This causes the active genetic material to converge, but the inactive genetic material to randomly *drift*. This results in a population with similar active genes but differing inactive genes. As mutation (or crossover) can result in inactive genes becoming active, the number of solutions one generation away is therefore much larger, and more dynamic, than without the inactive genes. For this reason inactive genes are thought to be beneficial through the process of NGD.

Another way of looking at the influence of NGD is through the effect redundant genes have on the search space. As redundant genes, by definition, do not affect the fitness, they create plateaus of equal fitness in the search space. These plateaus can be randomly *drifted* across when mutations of inactive/redundant genes occur. Depending upon the position on the plateaus, different solutions may be possible via mutation. For instance an inactive node may produce behaviour (a) if made active in one area of the search space or behaviour (b) if made active in another area of the search space. Again this can be seen to cause the possible solutions one mutation away to vary when trapped in a local optima.

Since CGP typically uses a $(1+\lambda)$-ES, all the children are created by mutating the single selected parent. Usually using such a greedy strategy would result in a search becoming very easily trapped in local optima. However, as we have discussed

CGP contains inactive genes which are subject to NGD. When in a local optima, any children created from the parent via mutations to inactive genes alone will have the same fitness as the parent; as they are semantically identical. An important aspect of CGP is that children are selected over parents if they have equal fitness. Therefore the solution selected may have the same fitness, but different inactive genetic material. This causes the possible solutions sampled through mutation to change from generation to generation and this helps to escape the local optima.

Another behaviour of NGD is its ability to alter inactive genes even when not trapped in local optima. For instance when redundant genes are mutated at the same time as beneficial mutations are made to active genes. In this case the changes to the inactive genes are also passed on to the selected child. This results in the possible solutions one mutation away being different to what they would have been had the neutral mutation not taken place. This effectively means that the plateaus in the search space can be followed even when not trapped in a local optima.

As the genotypes of other forms of GP also contain implicit genetic redundancy they too benefit from INGD. Likewise CGP genotypes also contain implicit genetic redundancy and so also benefit from INGD.

In the wider field of EC it is often argued that the benefit of genetic redundancy is related to its ability to protect from damaging mutations [29, 44, 47]. For instance when using a (μ, λ)-ES each new generation is entirely comprised of children; no parents. If the mutation rate were sufficiently high then the majority of the children would be drastically different from their parents. In this case good solutions may be completely lost from one generation to the next. Mutations to inactive genes do not change the phenotypes semantics. Therefore inactive genes help regulate the amount of semantic altering mutations. This is thought to help prevent good solutions from being so easily lost. However using redundant genes to absorb excess mutations appears to be equivalent to using a lower mutation rate [20]. It may also be the case that a varying level of redundant genes results in a varying number of mutations. However this to could be achieved using a different mutation method; for instance variable mutation. Regardless of whether this aspect of genetic redundancy is beneficial, CGP typically uses a $(1 + \lambda)$-ES so there is no danger of losing the best solution from one generation to the next. Additionally, in the experiments reported in this paper probabilistic mutation is used; which already allows a variable number of mutations to occur.

3.2 Increasing Genetic Redundancy

It has been previously reported in [27] that high levels of explicit genetic redundancy (95%) produce the best evolutionary search for CGP. In this work the percentage of inactive nodes was controlled by varying the number of available nodes. As CGP typically uses a fraction of the available nodes, increasing the number of available nodes has the effect of increasing the percentage of inactive nodes (genetic redundancy). The work showed that by increasing the number of available nodes the percentage of active nodes decreased and the effectiveness of the evolutionary search increased. The paper concluded that the increased genetic redundancy was responsible for the improved evolutionary search. Unfortunately, the paper did not expose the full relationship between the effectiveness of the

evolutionary search and the percentage of active nodes. For instance, from the results one could be left with the impression that using more available nodes always decreased the time to convergence. However, intuitively this cannot be true since it implies that using an unbounded number nodes would solve a given task in the first generation; a clear falsehood since the initial population is randomly initialized. Therefore, intuitively, there should be a point at which increasing the number of nodes no longer improves the evolutionary search.

It has also been shown that CGP is naturally biased towards phenotypes of a given size [14,15]; typically a small percentage of the available nodes. Although this does result in high levels of explicit genetic redundancy, it was shown that the active nodes are concentrated towards the inputs (low nodes indexes) and the inactive nodes towards the outputs (high nodes indexes). This results in few redundant genes *between* the active genes limiting the possible benefits of the redundant genes. It was also shown [14,15] that rearranging the active nodes, so as to increase the number of inactive nodes between the active nodes, strongly improved the evolutionary search. Thus demonstrating the importance of having the inactive nodes between the active. Additionally, it was shown that by rearranging the nodes, so as to increase the number of inactive nodes between active nodes, resulted in CGP more evenly sampling the solution space; in terms of the number of active nodes. Therefore removing the length bias. It was concluded that this less biased search was more effective.

It was also speculated in [14,15] that using very high numbers of available nodes, such as in [27], may be effective because it compensates for the lack of inactive nodes between the active. As using high numbers of nodes increases the likelihood of there being inactive nodes between the active. Therefore an increased fitness is seen as the number of nodes available to evolution is increased.

4 Benchmark Problems

The criteria for the choice of benchmarks are that they should be challenging enough to draw out any potential benefits of NGD and also represent a range of typical GP applications. The challenging aspect is significant as the investigations will be related to the escape from local optima. Therefore the tasks must be challenging enough that many local optima are encountered during the search. The range of applications selected comprise control, boolean logic and symbolic regression tasks.

For the experiments presented the CGP parameters are those given in Table 1 unless otherwise stated. These parameters are relatively "off-the-shelf" CGP parameters. As can be seen in Table 1, one hundred nodes are used. Although it is likely using a larger number would improve the results, using fewer makes the benchmarks more challenging and so only one hundred are used here. Additionally it can be seen that all experiments are repeated over 50 runs so reasonable averages can be used.

Where appropriate, the use of non-parametric statistical significance testing is used to assess any differences seen in the results. Here the Mann-Whitney U-test is used to test for statistical significance; with $\rho \leq 0.05$ representing statistical significance. Additionally the effect-size measure, as defined in [45], is also used to

Table 1: Default parameters.

Parameter	Value
Evolutionary Strategy	(1+4)-ES
Max Generations	100000
Num Runs	50
Mutation Scheme	probabilistic
Mutation Rate	3%
Maximum Number of Nodes	100
Node Arity	2

indicate the importance of any statistical difference; with values > 0.56 indicating a small effect size, > 0.64 a medium and > 0.71 a large.

4.1 Double Pole Balancing

The double pole balancing benchmark [48] is a popular challenging control task [16] of balancing two poles upon a movable cart; see Figure 3.

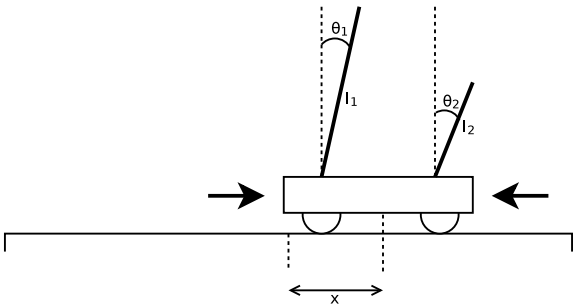


Fig. 3: Depiction of the double pole balancing benchmark.

The equations which describe the dynamics of the pole-cart system are given in Equations 2, 3, 4 and 5 with the symbol definitions and parameter values given in Table 2.

$$[ht]\ddot{x} = \frac{F - \mu_c sgn(\dot{x}) + \sum_{i=1}^{N} \tilde{F}_i}{M + \sum_{i=1}^{N} \tilde{m}_i} \tag{2}$$

$$\ddot{\theta}_i = -\frac{3}{4l_i}\left(\ddot{x}\cos\theta_i + g\sin\theta_i + \frac{\mu_{pi}\dot{\theta}_i}{m_i l_i}\right) \tag{3}$$

$$\tilde{F}_i = m_i l_i \dot{\theta}_i^2 \sin\theta_i + \frac{3}{4} m_i \cos\theta_i \left(\frac{\mu_{pi}\dot{\theta}_i}{m_i l_i} + g\sin\theta_i\right) \tag{4}$$

$$\tilde{m}_i = m_i\left(1 - \frac{3}{4}\cos^2\theta_i\right) \tag{5}$$

Table 2: Pole balancing parameter values.

Symbol	Description	Value
x	Cart position	$[-2.4, 2.4]m$
θ_i	i^{th} pole angle	$[-n, n]deg$
F	Force applied to cart	$[-10, 10]N$
$\tilde{F}_i$	Force on cart due to i^{th} pole	
l_i	Half length of i^{th} pole	$l_1 = 0.5m$ $l_2 = 0.05m$
M	Cart mass	$1.0kg$
m_i	Mass of i^{th} pole	$m_1 = 0.1kg$ $m_2 = 0.01kg$
μ_c	Friction coefficient between cart and track	0.0005
μ_{pi}	Friction coefficient between i^{th} pole and cart	0.000002

The inputs to the evolved controller are the position and velocity of the cart and the angle and angular velocities of the poles; making six inputs in total. These inputs are scaled linearly into a [-1,1] range assuming the following maximum values: cart position $[-2.4, 2.4]m$, cart velocity $[-1.5, 1.5]m/s$, pole angles $[-36, 36]deg$ and angular pole velocities $[-115, 115]deg/s$.

The single output of the controller determines the force applied to the cart. This output of the controller is capped to [-1,1] range and then linearly scaled to a [-10,10] N range before being applied to the cart. Additionally the force applied is made to be always greater than $\frac{1}{256} \times 10$N; this prevents the poles from ever being perfectly balanced thus increasing the difficulty of the task.

The cart is initially positioned in the centre of the track; with the longer and shorter poles starting at 1° and 0° from vertical respectively. The dynamics of the pole-cart system are then simulated using Euler integration with a time step of $0.01sec$ and the controller outputs are updated every $0.02sec$. The simulations are then ran for $100,000$ time steps.

The simulation is terminated when either the cart leaves the bounds of the track, the angle of either pole exceeds the maximum angle from vertical (36°) or a maximum of 100,000 time steps have elapsed. The fitness of each proposed solution is then 100,000 minus the number of elapsed time steps before termination; resulting in a optimal fitness of zero.

4.2 8 Bit Even Parity

The even parity benchmark is the task of implementing an even parity bit generator for a given number of specified bits. The benchmark can be extended to any number of bits with the difficulty of the tasks increasing as the number of bits increases. Here eight bit parity is used. In all cases the fitness awarded is the number of incorrect even parity bits generated after all possible inputs have been applied. Therefore the worse fitness is $256 = 2^8$. Here the function set contains AND, NAND, OR and NOR logic gates; the XOR and XNOR gates are deliberately excluded to increase the difficulty of the task.

4.3 4 Bit Full Adder

The full adder benchmark is the task of implementing a full adder circuit, utilising multiple outputs, for two bit strings of a specified number of bits. The difficulty of the task can be varied by increasing the number of bits to be added. Here a four bit full adder is used as it is reasonably challenging. The function set used contains AND, NAND, OR and NOR logic gates. The XOR and XNOR gates are deliberately excluded to increase the difficulty of the task.

The fitness assigned to each solution is the number of incorrect output bits produced after all possible inputs are applied. In the case of the four bit full adder there are nine inputs (two lots of four bits strings and one carry bit) and five outputs (enough to store the maximum possible sum). Therefore the worse possible fitness is $2560 = (2^9) \times 5$.

4.4 4 Bit Multiplier

The multiplier benchmark is the task of implementing a multiplier circuit, utilising multiple outputs, for two bit strings of a specified number of bits. The difficulty of the task can be varied by increasing the number of bits to be multiplied. Here a four bit multiplier is used as it is reasonably challenging. The function set used contains AND, NAND, OR and NOR logic gates. The XOR and XNOR gates are deliberately excluded to increase the difficulty of the task.

The fitness assigned to each solution is the number of incorrect output bits produced after all possible inputs are applied. In the case of the four bit multiplier there are eight inputs (two lots of four bits strings) and eight outputs (enough to store the maximum possible value). Therefore the worse possible fitness is $2048 = (2^8) \times 8$.

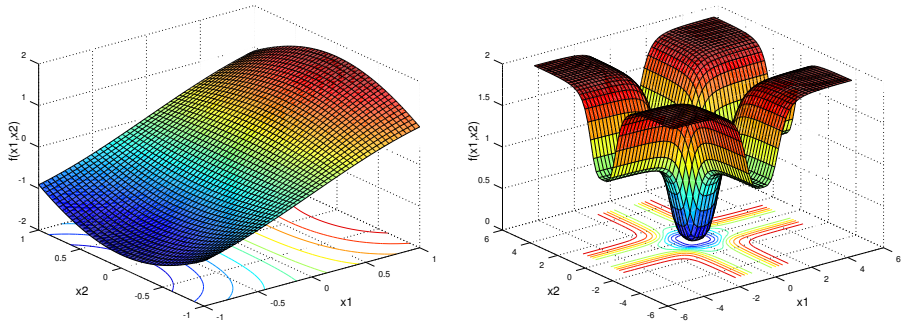
4.5 Nguyen 10

The Nguyen 10 benchmark is a symbolic regression task defined and used in [43]; given in Equation 6 and plotted in Figure 4. The task uses 100 random samples taken from the range x_1 in [-1,1] and x_2 in [-1,1]. The function set contains: $+, -, \times, \div, \sin, \cos, \exp, \log$. The fitness function aims to minimise the sum of the absolute errors between the correct value at a given x_1 and x_2 and the value produced by the evolved symbolic equation.

$$f(x_1, x_2) = 2\sin(x_1)\cos(x_2) \tag{6}$$

4.6 Pagie

The Pagie benchmark is a challenging [24] symbolic regression task defined and used in [32]; given in Equation 7 and plotted in Figure 4. The task uses 676 samples evenly taken from the range x_1 in [-5,5] and x_2 in [-5,5]. The function set contains: $+, -, \times, \div$. The fitness function is defined to be the sum of the absolute errors between the correct value at a given x_1 and x_2 and the value produced by



(a) Nguyen 10 Function (b) Pagie Function

Fig. 4: Nguyen 10 and Pagie Functions

the evolved symbolic equation. Thus the objective is to minimise the fitness (zero being the perfect score).

$$f(x_1, x_2) = \frac{1}{1 + x_1^{-4}} + \frac{1}{1 + x_2^{-4}} \tag{7}$$

4.7 Tower Problem

The Tower Problem [21] is a very challenging real world symbolic regression task. The data was provided by Arthur Kordon and gives 25 recordings of temperatures, flows, and pressures related to a distillation tower. The task is to create a symbolic equation which models the propylene concentration in the distillation tower based on these inputs.

The dataset contains 4999 samples (each containing 25 separate measurements and the propylene concentration) taken 15 minutes apart from within the distillation tower. The fitness awarded is the sum of the absolute error of the output after all inputs have been applied.

The function set used contains: $+, -, \times, \div, \sin, \cos, \exp, \log$.

5 Investigating Neutral Genetic Drift

Investigating INGD is challenging in tree-based GP due to implicitly redundant genes affecting only the semantics of the phenotype. For instance in Figure 1 the addition of y and z is only redundant because it is then later multiplied by zero. Although likely possible, detecting all instance of this type of redundancy would be very difficult and computationally expensive. Alternatively, investigating the role of explicit genetic redundancy present in CGP is much simpler. This is because the explicitly inactive genes can easily be identified; they are the genes not associated with the active nodes connecting inputs to outputs. It is this difference which is central to how CGP can be used to investigate NGD in general.

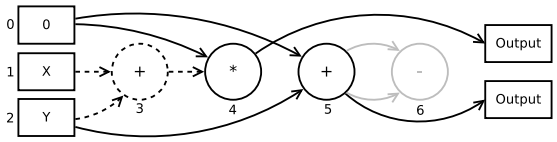
As previously discussed, NGD can be separated into two forms, namely INGD and ENGD. Additionally NGD poses two possible advantages/behaviours, the

ability to help escape local optima by following plateaus in the search space and the ability to follow the same plateaus simultaneously with positive mutations to active genes. Interestingly, these two behaviours are related to when neutral mutations are passed from parents to children. For instance, in order for NGD to help escape local optima, neutral mutations must be able to be passed on when the child's fitness is equal to that of its parents. Additionally for NGD to be able to change the inactive genes alongside positive mutations, neutral mutations must be passed on when the child's fitness is greater than its parents. Therefore the behaviour of NGD, implicit or explicit, is determined by *when* neutral mutations are passed on.

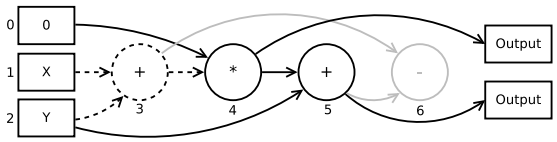
This means that there are four possible behaviours of NGD in total: ENGD occurring with fitness improvements, ENGD occurring without fitness improvements, INGD occurring with fitness improvements and finally INGD occurring without fitness improvements. These four behaviours are discussed in more detail for clarity.

In all the related figures presented, black represents active nodes, grey represents explicitly inactive nodes and dashed represents implicitly inactive nodes. Additionally (a) represents a possible parent and (b) represents a possible child which would have been selected over the given parent. The chromosomes of parent and child are also given with the function genes underlined and the differences highlighted in bold.

Figure 5 depicts an example of ENGD taking place at the same time as a fitness improving mutation. As can be seen in Figure 5, the child differs from the parent in two regards, both an explicitly inactive gene and an active gene have been mutated. This has resulted in a fitter solution which was why it was selected and therefore why the neutral mutation was passed on to the next generation.



(a) Parent: $\underline{0}12\ \underline{2}03\ \underline{0}02\ \underline{1}55\ 4\ 5$, fitness $= x$

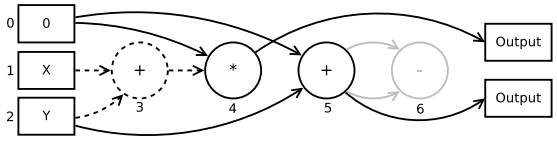


(b) Selected Child: $\underline{0}12\ \underline{2}03\ \underline{\mathbf{0}}\mathbf{4}2\ \underline{\mathbf{1}}\mathbf{3}5\ 4\ 5$, fitness $> x$

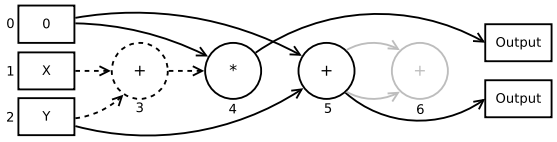
Fig. 5: Example of ENGD occurring with a fitness improvement. Active genes are given in bold, explicitly inactive in grey and implicitly inactive in dashed.

Figure 6 depicts ENGD taking place without a fitness improvement. In this case the selected child differs from the parent in one mutation to an explicitly inactive gene; marked again in bold in the chromosome. This mutation resulted in a child of equal fitness to the parent which is then selected because it was not

worse than the parent. This behaviour is only possible when children are selected over their parents when they have equal fitness. This aspect of NGD is thought to help the escape of local optima.



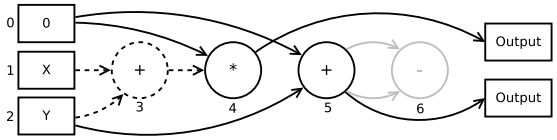
(a) Parent: $\underline{0}12\ \underline{2}03\ \underline{0}02\ \underline{1}55\ 4\ 5$, fitness $= x$



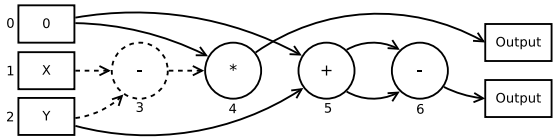
(b) Selected Child: $\underline{0}12\ \underline{2}03\ \underline{0}02\ \mathbf{\underline{0}}55\ 4\ 5$, fitness $= x$

Fig. 6: Example of ENGD occurring without a fitness improvement. Active genes are given in bold, explicitly inactive in grey and implicitly inactive in dashed.

Figure 7 depicts INGD taking place at the same time as an improving mutation. In this case the child differs from the parent due to one mutation to an implicitly inactive gene and one mutation to an active gene. These mutations resulted in a child of greater fitness than the parent which is why it was selected. In this case it can also be seen that the active mutation resulted in inactive genes becoming active. However this does *not* constitute NGD as no explicitly neutral genes have changed. Making inactive genes active is not an instance of NGD as no neutral genetic material has changed.



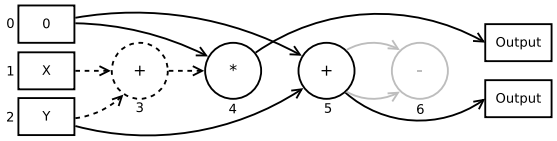
(a) Parent: $\underline{0}12\ \underline{2}03\ \underline{0}02\ \underline{1}55\ 4\ 5$, fitness $= x$



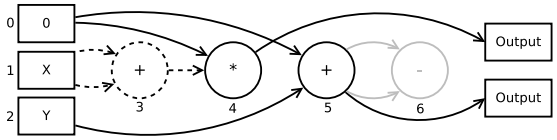
(b) Selected Child: $\mathbf{\underline{0}}11\ \underline{2}03\ \underline{0}02\ \underline{1}55\ 4\ \mathbf{6}$, fitness $> x$

Fig. 7: Example of INGD occurring with a fitness improvement. Active genes are given in bold, explicitly inactive in grey and implicitly inactive in dashed.

Finally, Figure 8 depicts INGD taking place without a fitness improvement mutation. In this case the child differs from the parent due to one mutation to an implicitly inactive gene. Again in this case the child was selected over the parent because it had equal fitness. This behaviour of INGD is thought to help escape local optima.



(a) Parent: $\underline{0}12\ \underline{2}03\ \underline{0}02\ \underline{1}55\ 4\ 5$, fitness $= x$



(b) Selected Child: $\underline{0}\mathbf{1}\mathbf{1}\ \underline{2}03\ \underline{0}02\ \underline{1}55\ 4\ 5$, fitness $= x$

Fig. 8: Example of INGD occurring without a fitness improvement. Active genes are given in bold, explicitly inactive in grey and implicitly inactive in dashed.

Now the four behaviours of NGD have been defined, methods for isolation are introduced. These methods are summarised in Table 3 for reference. The first method used for comparison is regular CGP; CGP with no alterations. Regular CGP is listed in Table 3 where it can be seen that it exhibits all of the behaviours of NGD discussed.

Table 3: Restrictions made to CGP used in order to investigate NGD.

Behaviour	Regular CGP	Only Mutate Active Genes	Only Fitness Improvements	Only Mutate Active Genes and Only Fitness Improvements
ENGD with fitness improvement	Yes	No	Yes	No
ENGD without fitness improvement	Yes	No	No	No
INGD with fitness improvement	Yes	Yes	Yes	Yes
INGD without fitness improvement	Yes	Yes	No	No

The second method, also given in Table 3, is to only mutate active genes; never inactive. This methods is the same as standard CGP with the one exception, explicitly inactive genes are never mutated. This is easily achieved for CGP as explicitly inactive genes are those genes not associated with the path of nodes connecting inputs to outputs. Once mutations obey this constraint both behaviours of ENGD

are prevented but both behaviours of INGD are allowed. This is because it completely prevents any change to explicitly inactive genes. However it is important to note that this does not prevent active genes connecting to inactive genes and thus making them active; which does not constitute NGD.

The third method, again given in Table 3, is to only select fitness improvements. This methods is the same as standard CGP except that children are only selected over their parents *iff* they have *greater* fitness; not equal or less. This alteration therefore prevents both ENGD and INGD occurring unless it is accompanied by a fitness improving mutation. Only selecting child chromosomes with fitness greater than their parents prevents NGD, implicit and explicit, from aiding the escape from local optima. It does however leave the possible benefit of NGD taking place alongside fitness improving mutations.

The final method combines the previous two described methods; only mutating active genes and only selecting fitness improvements. This removes all benefits of NGD, implicit and explicit, except specifically INGD taking place simultaneously with fitness improvements.

Using these described restrictions to CGP a series of experiments are proposed and presented which isolate various aspects of NGD.

As an aside it appears that by utilising similar methods it would also be possible to isolate whether INGD aids the escape from local optima. For instance in the case of equal fitness, if children were selected over their parents only when there were no mutations to active genes. However it is possible for a child to be created from a parent which, through mutation, represents a new solution of equal fitness. In this case the proposed method for preventing INGD would also prevent mutations not related to genetic redundancy. Additionally as there are typically many more inactive genes than active in CGP chromosomes, the chances of encountering INGD without ENGD are small. Therefore this proposed method prevents a large number of instances of INGD. It is not therefore a suitable method for isolating INGD aiding the escape of local optima.

5.1 Experiments

The first experiment is to isolate the overall benefit of ENGD. This is achieved by comparing the difference in performance between regular CGP and CGP when only active genes are mutated. These two methods differ only in the presence of ENGD and therefore isolate the benefit of ENGD. This comparison is given in Table 4 for the benchmarks described in Section 4.

As can be seen in Table 4, in all cases the presence of ENGD resulted in a more effective evolutionary search. Additionally in all but one case the difference resulted in a medium or large effect size with statistical significance. This clearly demonstrates that the presence of ENGD is greatly beneficial to the evolutionary search; at least in the case of CGP. It can therefore be concluded that the ability of ENGD to escape local optima, or the ability to mutate explicitly inactive genes at the same time as positive mutations, or both, is beneficial to the evolutionary search. Additionally as INGD is present in both of the methods used for comparison it can be seen that the benefit of ENGD is additional to that already provided by INGD.

Table 4: Comparing regular CGP to only mutating active genes in order to isolate the benefit of ENGD. In all cases a lower fitness represents a better search.

Benchmark	Regular CGP	Only Mutate Active Genes	U-Test	Effect Size
Double Pole	32517	58273	3.97E-3	0.65840
Full Adder 4 bit	352.92	400.06	1.16E-2	0.64660
Multiplier 4 bit	301.84	314.86	1.29E-2	0.64440
Nguyen10	0.78	1.60	3.88E-1	0.54780
Pagie1	73.13	105.97	1.32E-2	0.64400
Parity8Bit	45.08	68.22	1.01E-6	0.78380
Tower Problem	174404	241640	1.66E-11	0.89080

The second experiment investigates whether NGD, both implicit and explicit, provides an ability to aid the escape from local optima. This is achieved by comparing the difference in performance between regular CGP and CGP in which only fitness improvements are allowed to be selected. These two methods differ only in that the former (regular CGP) has the ability to pass on neutral mutations without the presence of beneficial mutations; a requirement for use in escaping local optima.

This experiment is similar to that undertaken in [26,46]; however there are important differences. For instance the range of benchmarks is much larger and statistical analysis is used to confirm any differences. Additionally in [26,46] this experiment was proposed to remove *all* benefits of NGD. However, as discussed, it only removes the benefit of NGD aiding the escape from local optima. For instance NGD can still occurs when neutral genes are mutated along with beneficial mutations to active genes. Here the results of the experiment are analysed with regard to only what is isolated.

As can be seen in Table 5, in all cases allowing NGD to escape local optima produces superior evolutionary search with statistical significance. Additionally in all but two cases this difference resulted in a large effect size; with the other two resulting in one medium effect size and one small. It can therefore be concluded that the ability of NGD to assist in escape from local optima is a major advantage; at least in the case of CGP.

Table 5: Comparing regular CGP to only selecting fitness improvements in order to isolate the benefit of NGD aiding the escape from local optima. In all cases a lower fitness represents a better search.

Benchmark	Regular CGP	Only Fitness Improvements	U-Test	Effect Size
Double Pole	32517	52225	2.62E-2	0.61920
Full Adder 4 bit	352.92	698.66	1.83E-17	0.99360
Multiplier 4 bit	301.84	402.68	4.05E-17	0.98820
Nguyen10	0.78	1.77	6.42E-3	0.65280
Pagie1	73.13	152.56	1.21E-8	0.83080
Parity8Bit	45.08	117.98	6.89E-18	1
Tower Problem	174404	218169	4.58E-12	0.90160

The third experiment investigates the benefit of INGD solely as a means of escaping from local optima. For instance the benefit of INGD with and without it occurring at the same time as beneficial mutations. This is achieved by comparing the difference between only mutating active genes and only mutating active genes whilst also only allowing fitness improvements to be selected. For clarity this is a comparison between the methods in the third and fifth columns of Table 3. These two methods differ only in the ability of INGD to occur without improving mutations and so isolates this behaviour.

As can be seen Table 6, allowing INGD to aid the escape of local optima always resulted in a superior evolutionary search. Additionally in the majority of cases the differences was statistically significant with a large effect size. However in two cases the difference was not statistically significant and the effect size was small. It is therefore not always offering an advantage large enough to be of importance.

Table 6: Mutating active genes compared to only allowing mutations to active genes with fitness improvements in order to isolate the benefit of INGD escaping local optima. In all cases a lower fitness represents a better search.

Benchmark	Only Mutate Active Genes	Only Mutate Active Genes, Only Fitness Improvements	U-Test	Effect Size
Double Pole	58273	69017	7.82E-2	0.60120
Full Adder 4 bit	400.06	745.36	1.03E-17	0.99740
Multiplier 4 bit	314.86	410.04	7.95E-17	0.98360
Nguyen10	1.60	4.70	8.51E-5	0.72440
Pagie1	105.97	175.88	2.05E-5	0.74720
Parity8Bit	68.22	120.80	6.98E-18	0.99960
Tower Problem	241640	260060	7.51E-2	0.60320

The fourth experiment investigates the benefit of ENGD when not being used to escape local optima. For instance the benefit of ENGD occurring at the same time as other beneficial mutations. This is achieved by comparing the difference between only allowing fitness improvements to be passed on and only allowing mutations to active genes whilst also only allowing fitness improvements to be passed on. For clarity this is a comparison between the methods in the final two columns of Table 3. These two methods differ only in the ability of ENGD to occur in combination with fitness improving mutations to active genes.

As can be seen in Table 7, removing the benefit of ENGD occurring together with other beneficial mutations consistently produced inferior results with statistical significance in all but one case. However in the majority of cases, all but two, the effect size was small. Therefore it appears that although ENGD does have benefit other than merely assisting in the escape from local optima, this benefit is small.

6 Investigating Increasing Explicit Genetic Redundancy

As has been discussed in Section 3, it has been previously presented that CGP produces a more effective search as increased numbers of available nodes are used [27]. The explanation given was that using more available nodes increased the level

Table 7: Selecting only fitness improvements compared with only allowing mutations to active genes while only selecting fitness improvements. This isolates the benefits of ENGD other than aiding the escape of local optima. In all cases a lower fitness represents a better search.

Benchmark	Only Fitness Improvements	Only Mutate Active Genes, Only Fitness Improvements	U-Test	Effect Size
Double Pole	52225	69017	3.30E-2	0.62160
Full Adder 4 bit	698.66	745.36	3.33E-2	0.62380
Multiplier 4 bit	402.68	410.04	4.46E-1	0.54440
Nguyen10	1.77	4.70	7.11E-3	0.65480
Pagie1	152.56	175.88	1.72E-1	0.57940
Parity8Bit	117.98	120.80	2.74E-2	0.62780
Tower Problem	218169	260060	1.90E-6	0.77640

of explicit genetic redundancy which lead to a more effective evolutionary search. However an alternative explanation was later given arguing that the additional genetic redundancy would be unlikely to be ever used and the benefit was due to the high levels of available nodes compensating for length bias [14, 15].

Regardless of the explanation it still stands that using an increased number of available nodes was seen to improve the evolutionary search for CGP. Additionally as previous work did not show the full trend of this relationship (i.e. it did not allow large enough genotypes) it is difficult to draw further conclusions. Once the full trend is identified it may be possible to to distinguish between the two explanations as to why allowing larger genotypes is more effective.

6.1 Experiments

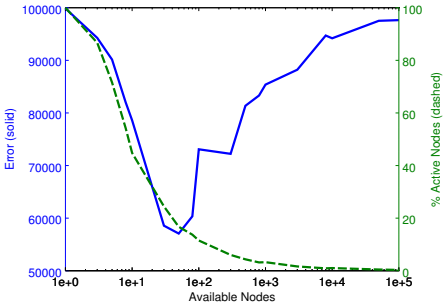
The experiment is an extension of previously presented work [27]. The work extends previous work in a number of regards. The number of benchmarks is much larger, the range of problem types is larger, the benchmarks are much more challenging and, most importantly, the range of the number of available nodes is much larger.

For the experiments presented the same benchmarks as described in Section 4 are used. The CGP parameters are assumed from Table 1 with the exceptions that 1) the maximum number of generations was reduced to 10,000[3], and 2) the number of available nodes investigated are in the range one to one hundred thousand; note previous research studied the relationship up to 4,000 available nodes [27].

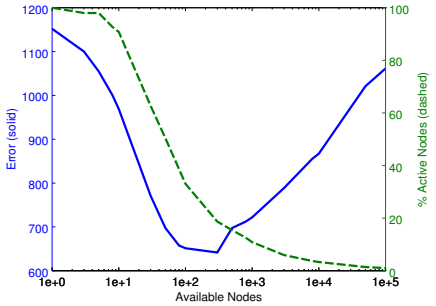
The results of varying the number of available nodes are given in Figure 9 for all of the benchmarks described. Each of the individual plots show the fitness achieved and the percentage of active nodes vs the number of available nodes; note the logarithmic scale on the x-axis. Both fitness and percentage of active nodes are presented so the relationship between the levels of explicit genetic redundancy and the effectiveness of the evolutionary search can be compared.

A number of interesting features can be identified in Figure 9. Firstly, consider the fitness achieved under variation in the number of available nodes. It can be seen that in each case the fitness first improves as the number of available nodes is

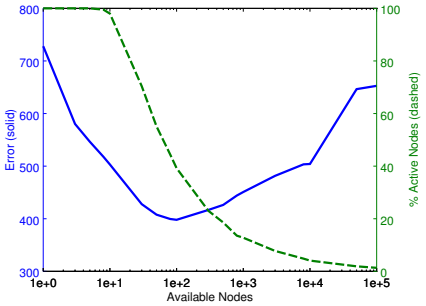
[3] The reduction in generations was to reduce the computational expense of the experiments.



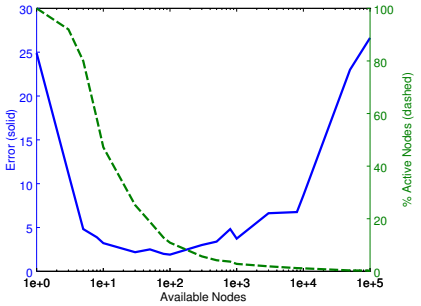
(a) Double Pole



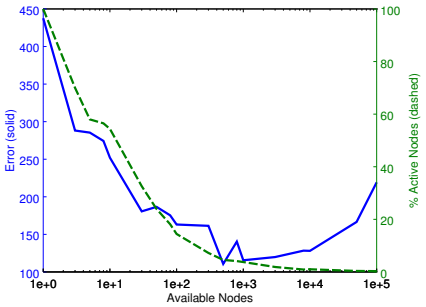
(b) Full Adder 4 bit



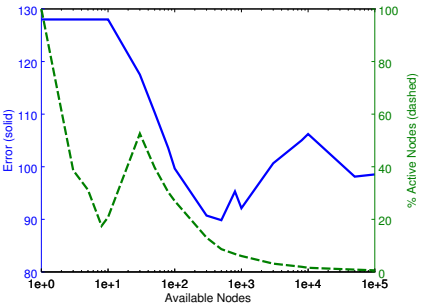
(c) Multiplier 4 bit



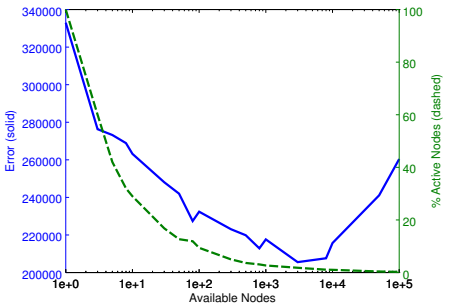
(d) Nguyen 10



(e) Pagie1



(f) Parity8Bit



(g) Tower Problem

Fig. 9: Increasing the number of available nodes Vs Fitness and percentage of active nodes.

increased until a optimum is reached and then the fitness worsens as the number of available nodes continues to increase.

In the case of the percentage of available nodes the general trend is to start at approximately 100% and then continuously fall, following an approximate sigmoid curve, as the number of available nodes is increased; asymptotically approaching zero percent.

The fact that low numbers of available nodes results in a high percentage of active nodes is intuitive as in this case the benchmarks are likely to require more nodes than are available. Under these circumstances there is likely an evolutionary pressure to use as many nodes as possible; in order to best solve the give task.

It can also be seen that initially the fitness achieved is closely correlated with the percentage of active nodes. This accounts for the conclusions reached in [27]. However past the optimal number of available nodes the percentage of active nodes continues to fall but the achieved fitness *worsens*. That is to say it is not the case that increasing the percentage of inactive nodes improves the evolutionary search for CGP. Nor is it that case that increasing the number of available nodes improves the evolutionary search for CGP. It appears that the effectiveness of the evolutionary search is determined by the number of available nodes but this is not correlated with the percentage of inactive nodes.

7 Discussion

This section gives further discussion of the results presented in Sections 5 and 6. The results are separated into the individual investigations.

7.1 Neutral Genetic Drift

From the experiments presented four properties of NGD in CGP have been identified. It has been shown that ENGD is greatly beneficial to CGP. It has been shown that the ability for NGD to escape local optima represent the most beneficial aspect of NGD for CGP. It has also been shown that the ability of INGD specifically to escape local optima is beneficial. And finally it has been shown that ENGD occurring when not trapped in local optima also offers a benefit.

For many years it has been thought that ENGD is highly beneficial to CGP. However, previous work investigating the effect of ENGD did so by preventing both INGD and ENGD from escaping local optima [26,46]. Therefore previous research did not isolate the effect of ENGD nor did it consider benefits other than escaping local optima. In the work presented here the role of specifically ENGD has been isolated and shown to be of great benefit. Although this means that the hypothesis that ENGD provides a benefit remains the same, the evidence for believing this has changed.

It has also been shown here that ENGD provides a small benefit to the evolutionary search when taking place along side positive mutations to active genes. This is an aspect of ENGD not previously considered in CGP. A possible explanation for this benefit could lie in the effect it has on the positioning in the search space. Mutating inactive genes along with active genes could cause the semantics of the solution to alter only slightly but place the solution in a very different area

of the search space; in terms of what solutions are reachable in one generation. Although highly speculative, this could be causing a wider area of the solution space to be sampled during the evolutionary search; as it is not limited to the solutions which can only be reached (or easily reached) via successive fitness improving mutations to active genes.

An important insight from the experiments presented is that it is now known that the benefits of ENGD are *additive* to INGD; at least in CGP. This was shown to be the case as preventing ENGD, whilst preserving INGD, produced a worse evolutionary search. Additionally it was seen that benefits provided by INGD escaping local optima were less than the benefits provided by both INGD and ENGD escaping local optima. This is important as it is now known that the presence of specifically explicit genetic redundancy is advantageous to the evolutionary search. Therefore other methods which contain explicit genetic redundancy, such as Linear GP [4], Grammatical Evolution (GE) [30,31,35] and Push-GP [37] could also be benefiting from ENGD. Interesting future work would be to repeat the experiment presented here using other GP methods which contain explicit genetic redundancy. This would identify if the results are general or specific to CGP. In addition, the work suggests that if explicitly neutral genes could be added to other GP methods, it could lead to a more effective search. For instance, if "switch genes" were introduced on links between nodes in GP trees that could connect or disconnect sub-trees, then all genes in switched off sub-trees would become explicitly neutral. This could enable tree-based GP to benefit from ENGD.

The fact that the addition of ENGD to CGP was shown to provide a benefit indicates that the amount of genetic redundancy provided implicitly is not sufficient to fully utilise NGD. Therefore it may be the case that other GP methods, such as tree-based GP, do not fully utilise NGD. Although the amount of useful explicit genetic redundancy present in CGP is unlikely to be optimal, it is providing a further benefit.

Interestingly it could be the case that increasing or decreasing the amount of mutation to explicitly inactive genetic material would improve the evolutionary search of CGP. For instance, it has been shown in this paper that never mutating explicitly inactive genes significantly worsens the evolutionary search. This method is equivalent to an explicitly inactive gene mutation rate of 0%. When mutations were allowed to alter explicitly inactive genetic material the same mutation rate was used as for the active genes (i.e. 3%). However there is no reason to assume that this is a suitable mutation rate to make best use of explicit genetic redundancy. For instance it could be the case that a much larger mutation rate for inactive genes typically produces better results, possibility even 100% i.e. completely random. Additionally it may be the case that different explicitly inactive genetic mutation rates produce the best search depending upon whether the search is trapped in a local optima.

A related topic is whether partial solutions are maintained within the genetic redundancy ready to be reactivated at a later stage. For instance on dynamic fitness landscapes a reasonable solution could be found to no longer be suitable. This may lead to sections of the chromosome becoming inactive whilst looking for a new suitable solution. Now if the fitness landscape continues to change then the old solution, still present in the inactive genetic material, may once again be beneficial. However this time, instead of completely rediscovering the solution, the old solution can be reactivated ('remembered'). Whether this would actually oc-

cur is speculative, and it has been shown that CGP rarely activates a previously active node without mutation [14]. However it does demonstrate additional possible behaviours of explicitly inactive genes, other than NGD, which could also be manipulated.

The fact that NGD, both implicit and explicit, aid the escape from local optima is an important result as it indicates that it may be increasingly beneficial on more challenging tasks. This is because one of the reasons a given task is more challenging than another is due to the number local optima and how difficult they are to escape from. Therefore methods which aid the escape of local optima are likely to be of greater benefit on more challenging fitness landscapes. Future work should therefore investigative applying CGP to increasingly difficult problem instances by repeating the experiments presented here. This would identify if NGD does become increasingly beneficial on harder tasks.

One of the reasons the research presented here was possible was due to the ease of which explicitly inactive genes in CGP can be identified. This makes the study of NGD much simpler. Additionally if it were found that certain characteristics were desirable in inactive genetic material, such as specific mutation methods, this could be easily implemented for explicit genetic redundancy. Therefore although many methods, such as tree-based GP, do contain implicit genetic redundancy, empirically studying its effect is harder. It remains for future investigation whether there is an effective method of identifying implicitly inactive genetic redundancy in tree-based GP. Notwithstanding this, the speed at which explicitly inactive genetic redundancy can be identified may still be important.

7.2 Increasing Explicit Genetic Redundancy

From Figure 9 it can be seen that increasing the number of available nodes does cause the percentage of active nodes to decrease. Additionally increasing the number of available nodes also *initially* improves the evolutionary search for CGP. This is as was reported in [27] and what lead to their conclusions. However as the number of available nodes continues to increase the percentage of activate nodes continues to fall but the effectiveness of the evolutionary search begins to *worsen*. This disproves the theory that the effectiveness of the evolutionary search is correlated with the number of available nodes.

It has been previously shown that CGP exhibits a length bias towards a given number of active nodes [14,15]. This length bias is a function of the number of inputs, the number of nodes, the arity of each node and the number of outputs. Therefore by varying the number of available nodes one is effectively varying the number of nodes to which there is a bias. As there is likely an optimal number of nodes for a given task it makes sense that as the number of nodes is varied the performance first improves to an optimum and then worsens. In this regard the number of available nodes is like many other evolutionary parameters whose optimal value is task dependent. Note that this result does not contradict the evidence for the benefit of explicit genetic redundancy. It is only that explicit genetic redundancy introduced by increasing the number of available nodes is not beneficial. As was described in [14,15], the majority of these inactive genes will describe nodes positioned towards the outputs (high node index) and will be very unlikely to ever be made active. That is to say, the level of explicit genetic redundancy is

increased, but in such a way that is is rarely used. Therefore increasing the number of available nodes is not an effective method of increasing the levels of genetic redundancy.

Additionally from the plots given in Figure 9 it does not appear that 95% genetic redundancy provides the best evolutionary search for CGP; as was proposed in [27]. So clearly this is problem dependent.

This result corrects a previous misconception that continuously increasing the number of available nodes is beneficial to CGP's evolutionary search [27]. Additionally this result, coupled with previous research [14,15], demonstrates that increasing the number of available nodes is not an effective method of increasing the levels of *useful* genetic redundancy in CGP. It would be interesting to investigate position dependent mutation schemes that are related to the probability of nodes being active. It might be possible to control the amount of redundancy and show whether it is correlated with the effectiveness of the evolutionary search.

8 Conclusion

The work presented has undertaken a detailed analysis of the role of NGD in CGP. Previous misconceptions have been corrected and further insights made.

It is now known that the levels of genetic redundancy provided implicitly is not sufficient to best utilise NGD in CGP. It has been shown that the presence of explicit genetic redundancy offers a significant further advantage. It is also likely that other forms of EC which contain explicit genetic redundancy are also benefiting from its presence. Additionally other forms of EC which do not contain explicit genetic redundancy may benefit from its inclusion.

It has also been shown that isolating and manipulating explicit genetic redundancy is far easier than controlling implicit genetic redundancy. This makes the study of explicit genetic redundancy much simpler and enables easy adaptation to make best use of its presence. For instance by varying its mutation rate to increase or decrease the rate of drift. This may lead to explicit genetic redundancy being more desirable than implicit if it can be fully utilised and controlled. This might be important as the difficulty of the search problems increases.

Additional benefits of NGD, other than escaping local optima, have also been empirically isolated and demonstrated to aid the evolutionary search.

References

1. Banzhaf, W.: Genotype-phenotype-mapping and neutral variation-a case study in genetic programming. In: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, pp. 322–332. Springer-Verlag (1994)
2. Barnett, L.: Ruggedness and neutrality: The nkp family of fitness landscapes. In: Artificial Life VI: Proceedings of the sixth international conference on Artificial life, pp. 18–27 (1998)
3. Blickle, T., Thiele, L.: Genetic programming and redundancy. choice **1000**, 2 (1994)
4. Brameier, M., Banzhaf, W.: Linear Genetic Programming. Springer (2007)
5. Clegg, J., Walker, J.A., Miller, J.F.: A new crossover technique for Cartesian Genetic Programming. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 1580–1587. ACM (2007)
6. Collins, M.: Finding needles in haystacks is harder with neutrality. Genetic Programming and Evolvable Machines **7**(2), 131–144 (2006)

7. Ebner, M.: On the search space of genetic programming and its relation to nature's search space. In: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, vol. 2. IEEE (1999)
8. Ebner, M., Langguth, P., Albert, J., Shackleton, M., Shipman, R.: On neutral networks and evolvability. In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, vol. 1, pp. 1–8. IEEE (2001)
9. Ebner, M., Shackleton, M., Shipman, R.: How neutral networks influence evolvability. Complexity **7**(2), 19–33 (2001)
10. Fonseca, C.M., Correia, M.B.: Developing redundant binary representations for genetic search. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on, vol. 2, pp. 1675–1682. IEEE (2005)
11. Forrest, S., Mitchell, M.: Relative building-block fitness and the building-block hypothesis (1993)
12. Galván-López, E., Dignum, S., Poli, R.: The effects of constant neutrality on performance and problem hardness in gp. In: EuroGP 2008, pp. 312–324. Springer (2008)
13. Galván-López, E., Poli, R., Kattan, A., ONeill, M., Brabazon, A.: Neutrality in evolutionary algorithms what do we know? Evolving Systems **2**(3), 145–163 (2011). DOI 10.1007/s12530-011-9030-5. URL http://dx.doi.org/10.1007/s12530-011-9030-5
14. Goldman, B., Punch, W.: Analysis of Cartesian Genetic Programmings Evolutionary Mechanisms. Evolutionary Computation, IEEE Transactions on **PP**(99), 1–1 (2014). DOI 10.1109/TEVC.2014.2324539. In press
15. Goldman, B.W., Punch, W.F.: Length bias and search limitations in Cartesian Genetic Programming. In: Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference, pp. 933–940. ACM (2013)
16. Gomez, F., Schmidhuber, J., Miikkulainen, R.: Accelerated neural evolution through cooperatively coevolved synapses. The Journal of Machine Learning Research **9**, 937–965 (2008)
17. Huynen, M.A.: Exploring phenotype space through neutral evolution. Journal of molecular evolution **43**(3), 165–169 (1996)
18. Huynen, M.A., Stadler, P.F., Fontana, W.: Smoothness within ruggedness: the role of neutrality in adaptation. Proceedings of the National Academy of Sciences **93**(1), 397–401 (1996)
19. Kimura, M., et al.: Evolutionary rate at the molecular level. Nature **217**(5129), 624–626 (1968)
20. Knowles, J.D., Watson, R.A.: On the utility of redundant encodings in mutation-based evolutionary search. In: Parallel Problem Solving from NaturePPSN VII, pp. 88–98. Springer (2002)
21. Kordon, A.: Tower problem (2015). URL http://www.symbolicregression.com/
22. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
23. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer-Verlag (2002)
24. McDermott, J., White, D.R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., et al.: Genetic programming needs better benchmarks. In: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference, pp. 791–798. ACM (2012)
25. Miller, J.F.: What bloat? Cartesian genetic programming on Boolean problems. In: 2001 Genetic and Evolutionary Computation Conference Late Breaking Papers, pp. 295–302 (2001)
26. Miller, J.F. (ed.): Cartesian Genetic Programming. Springer (2011)
27. Miller, J.F., Smith, S.: Redundancy and computational efficiency in Cartesian Genetic Programming. Evolutionary Computation, IEEE Transactions on **10**(2), 167–174 (2006)
28. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Proceedings of the Third European Conference on Genetic Programming (EuroGP), vol. 1820, pp. 121–132. Springer-Verlag (2000)
29. Nordin, P., Francone, F., Banzhaf, W.: Explicitly defined introns and destructive crossover in genetic programming. In: P.J. Angeline, K.E. Kinnear Jr. (eds.) Advances in Genetic Programming, pp. 111–134. MIT Press, Cambridge, MA, USA (1996). URL http://dl.acm.org/citation.cfm?id=270195.270205
30. O'Neill, M., Ryan, C.: Grammatical Evolution. IEEE Transactions on Evolutionary Computation **5**(4), 349–358 (2001)

31. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Springer (2003)
32. Pagie, L., Hogeweg, P.: Evolutionary consequences of coevolving targets. Evolutionary computation **5**(4), 401–418 (1997)
33. Poli, R., Langdon, W.W.B., McPhee, N.F., Koza, J.R.: A field guide to Genetic Programming. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk (2008)
34. Rothlauf, F., Goldberg, D.E.: Redundant representations in evolutionary computation. Evolutionary Computation **11**(4), 381–415 (2003)
35. Ryan, C., Collins, J., Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: W. Banzhaf, R. Poli, M. Schoenauer, T. Fogarty (eds.) Genetic Programming, *Lecture Notes in Computer Science*, vol. 1391, pp. 83–96. Springer Berlin Heidelberg (1998)
36. Silva, S., Costa, E.: Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. Genetic Programming and Evolvable Machines **10**(2), 141–179 (2009)
37. Spector, L., Robinson, A.: Genetic Programming and Autoconstructive Evolution with the Push Programming Language. Genetic Programming and Evolvable Machines **3**(1), 7 – 40 (2002)
38. Turner, A.J., Miller, J.F.: Cartesian Genetic Programming: Why No Bloat? In: Genetic Programming: 17th European Conference, EuroGP-2014, *LNCS*, vol. 8599, pp. 193–204. Springer-Verlag Berlin Heidelberg (2014)
39. Turner, A.J., Miller, J.F.: Introducing A Cross Platform Open Source Cartesian Genetic Programming Library. Genetic Programming and Evolvable Machines **16**(1), 83–91 (2014). DOI 10.1007/s10710-014-9233-1
40. Turner, A.J., Miller, J.F.: Recurrent Cartesian Genetic Programming. In: 13th International Conference on Parallel Problem Solving from Nature (PPSN 2014), *LNCS*, vol. 8672, pp. 476–486 (2014)
41. Turner, A.J., Miller, J.F.: Recurrent Cartesian Genetic Programming Applied to Famous Mathematical Sequences. In: Proceedings of the Seventh York Doctoral Symposium on Computer Science & Electronics, pp. 37–46 (2014)
42. Turner, A.J., Miller, J.F.: Recurrent Cartesian Genetic Programming Applied to Series Forecasting. In: Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO-15) (2015). To Appear
43. Uy, N.Q., Hoai, N.X., ONeill, M., McKay, R.I., Galván-López, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines **12**(2), 91–119 (2011)
44. Van Nimwegen, E., Crutchfield, J.P., Huynen, M.: Neutral evolution of mutational robustness. Proceedings of the National Academy of Sciences **96**(17), 9716–9720 (1999)
45. Vargha, A., Delaney, H.D.: A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics **25**(2), 101–132 (2000)
46. Vassilev, V.K., Miller, J.F.: The Advantages of Landscape Neutrality in Digital Circuit Evolution. In: Proc. International Conference on Evolvable Systems, *LNCS*, vol. 1801, pp. 252–263. Springer Verlag (2000)
47. Wagner, A.: Robustness, evolvability, and neutrality. FEBS letters **579**(8), 1772–1778 (2005)
48. Wieland, A.: Evolving neural network controllers for unstable systems. In: Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on, vol. 2, pp. 667–673. IEEE (1991)
49. Wilson, D., Kaur, D.: Search, neutral evolution, and mapping in evolutionary computing: A case study of grammatical evolution. IEEE Transactions on Evolutionary Computation **13**(3), 566–590 (2009)
50. Wright, S.: The Roles of Mutation, Inbreeding, Crossbreeding, and Selection in Evolution. In: Sixth International Congress of Genetics, pp. 356–366. Brooklyn Botanic Garden (1932)
51. Yu, T., Miller, J.: Neutrality and the evolvability of boolean function landscape. In: J. Miller, M. Tomassini, P. Lanzi, C. Ryan, A. Tettamanzi, W. Langdon (eds.) Genetic Programming, *Lecture Notes in Computer Science*, vol. 2038, pp. 204–217. Springer Berlin Heidelberg (2001)
52. Yu, T., Miller, J.: Finding needles in haystacks is not hard with neutrality. In: J. Foster, E. Lutton, J. Miller, C. Ryan, A. Tettamanzi (eds.) Genetic Programming, *Lecture Notes*

in Computer Science, vol. 2278, pp. 13–25. Springer Berlin Heidelberg (2002). DOI 10.1007/3-540-45984-7_2. URL http://dx.doi.org/10.1007/3-540-45984-7_2
53. Yu, T., Miller, J.F.: Through the interaction of neutral and adaptive mutations, evolutionary search finds a way. Artificial Life **12**(4), 525–551 (2006)