Genet Program Evolvable Mach (2009) 10:417–445
DOI 10.1007/s10710-009-9093-2

ORIGINAL PAPER

Parallel evolution using multi-chromosome cartesian genetic programming

James Alfred Walker · Katharina Völk · Stephen L. Smith · Julian Francis Miller

Received: 15 May 2009 / Revised: 28 August 2009 / Published online: 20 October 2009

Abstract Parallel and distributed methods for evolutionary algorithms have concentrated on maintaining multiple populations of genotypes, where each genotype in a population encodes a potential solution to the problem. In this paper, we investigate the parallelisation of the genotype *itself* into a collection of independent chromosomes which can be evaluated in parallel. We call this multi-chromosomal evolution (MCE). We test this approach using Cartesian Genetic Programming and apply MCE to a series of digital circuit design problems to compare the efficacy of MCE with a conventional single chromosome approach (SCE). MCE can be readily used for many digital circuits because they have multiple outputs. In MCE, an independent chromosome is assigned to each output. When we compare MCE with SCE we find that MCE allows us to evolve solutions much faster. In addition, in some cases we were able to evolve solutions with MCE that we unable to with SCE. In a case-study, we investigate how MCE can be applied to to a single objective problem in the domain of image classification, namely, the classification of breast X-rays for cancer. To apply MCE to this problem, we identify regions of interest (RoI) from the mammograms, divide the RoI into a collection of sub-images and use

J. A. Walker (✉) · K. Völk · S. L. Smith · J. F. Miller
Intelligent Systems Group, Department of Electronics, University of York,
Heslington, York YO10 5DD, UK
e-mail: jaw500@ohm.york.ac.uk

K. Völk
e-mail: kv500@ohm.york.ac.uk

S. L. Smith
e-mail: sls5@ohm.york.ac.uk

J. F. Miller
e-mail: jfm7@ohm.york.ac.uk

a chromosome to classify each sub-image. This problem allows us to evaluate various evolutionary mutation operators which can pairwise swap chromosomes either randomly or topographically or reuse chromosomes in place of other chromosomes.

Keywords Multiple chromosomes · Cartesian genetic programming · Digital circuits · Mammography · Parallelisation

1 Introduction

Parallel and distributed evolutionary algorithms have hitherto concentrated on techniques for maintaining a collection of sub-populations of genotypes, where each genotype is a complete encoding of a potential solution. In this paper we propose a different approach. We maintain a population of *sub-solutions* to the problem in hand, each of which can be evaluated independently. Each sub-solution is encoded in a *chromosome* and the genotype is a collection of chromosomes. We call it multi-chromosomal evolution (MCE).

We have evaluated this idea using a form of Genetic Programming (GP) called Cartesian Genetic Programming (CGP). This is a particularly suitable form of GP to tackle this investigation as CGP is well suited to problems having multiple outputs. MCE can be employed in problems where either multiple program outputs are required (e.g. multi-output digital circuits) or large problems can be subdivided into independent parts (e.g. spatial division of images into a collection of smaller image problems). We have investigated both these types of problems. The first type of problem we investigated was the evolution of multi-output combinational circuits. We compared the effectiveness of Multi-chromosome CGP with single chromosome CGP and show, in some cases, that we can solve problems with MCE which we could not solve with a single chromosome approach.

In a follow up to our comparative investigation of MCE and SCE, we report a case-study on the merits of MCE for locating potentially cancerous regions in mammograms. The single objective and spatial nature of images allows us to investigate and compare the effectiveness of a number of evolutionary mutation operators and evolutionary strategies. These allow chromosomes to be: swapped with their neighbours, swapped with other randomly chosen chromosomes, or even to replace other randomly chosen chromosomes. One of the additional benefits of MCE for this kind of problem is that one obtains a large number of CGP programs (in our case 256) that can be used in voting schemes to provide reliable and effective classification of malignancy.

The plan for the paper is as follows: Sect. 2 gives an overview of multi-chromosomal approaches in evolutionary computation and also explains the CGP method. In Sect. 3 we describe the multi-chromosome approach we have used for the evolution of digital circuits and the classification of malignancy in mammograms. The details of our experimental methodologies are given in Sect. 4. Experimental results are presented and analysed for digital circuits in Sect. 5. Section 6 describes experimental results and analysis for classification of mammograms. Section 7 provides conclusions and some suggestions for future work.

2 Background

2.1 Multiple chromosomes in genetic programming

Multiple chromosomes have been used in a number of ways within GP. One of the first was by Hillis [15], who co-evolved genotypes comprising fifteen pairs of chromosomes, to produce minimal sorting networks that were capable of outperforming human designs. Mayer and Spitzlinger [22] used a multi-chromosome genotype when investigating a biologically inspired, two-stage crossover operator, which combined a multi-point crossover with chromosome shuffling. The aim of the crossover operator was to produce offspring that inherit chromosomes from a number of parents in the population. Although the approach performed well, there was no overall advantage over a single chromosome approach using a 2-point crossover, on the optimisation problems tested. Cavill et al. [3–5] also discovered that the use of multiple chromosomes, and also having multiple copies of chromosomes within the representation is advantageous to evolution on symbolic regression problems. Using a two-stage crossover operator, similar chromosomes from two parents are paired using chromosome shuffling and then a *n*-point crossover is used to exchange material between the pairs of chromosomes.

Others have used multi-chromosomes for evolving genotype-phenotype mappings. Chow [6] used a representation divided into a binary-based data chromosome, and an integer-based mapping chromosome. The mapping chromosome contains a combination of the bit positions in the data chromosome, and is used to re-order the binary string of the data chromosome to produce the phenotype. This approach was shown to perform well on GA-hard and deceptive problems. Similarly, Corne et al. [10] used an approach which applied a multiploid genotype with a mapping chromosome to multiple knapsack and set covering problems. However, in this approach the mapping chromosome is used to determine which chromosome each gene in the expressed genotype is taken from. Both Chow and Corne et al. used a crossover operator which probabilistically exchanged genetic material between the chromosomes of two parents. Chow specified that the chromosomes must be of the same type [6], whereas Corne *et al* specified that the chromosomes were located at the same position in both genotypes [10].

If an approach does not explicitly mention the term “chromosome”, it does not necessarily mean that it is not a multi-chromosome approach. A good example of this is evolving a team of individuals who work together to solve a particular task [14, 20]. In this case, the genotype is divided into a number of regions, each encoding an individual. In such a situation, when there is only a single genotype, the regions are equivalent to chromosomes. Team-based approaches have also been used for robot soccer [2, 19, 33] but they have also been successfully applied to a range of problems not normally associated with teams, such as symbolic regression, which demonstrates the ability and value of multi-chromosome approaches [14, 35, 36]. Luke and Spector, and Haynes et al. have both experimented with two different crossover operators. The first allows the exchange of genetic material between any two chromosomes, located anywhere within each of the parents. The second is a restricted version of the first crossover operator, and is the same as that used by Corne et al., described in the previous paragraph. This restricted crossover operator was shown to perform equal to, or better than, the unrestricted crossover operator on team-based problems [20]. It was also adopted by Soule [35, 36].

Another approach which could be classed as multi-chromosome is Multi Expression Programming (MEP) [31]. In MEP, groups of statements (which could be seen as chromosomes) are evolved as linear genetic programs, where any statement may re-use a previous statement in the genotype in order to construct complex expressions. The output of the program does not come from a statement at a fixed point in the genotype, instead every statement is evaluated and the program output comes from the fittest statement. This approach has been successfully applied to simple adder and multiplier problems [30] and the knapsack problem [32]. In MEP, a uniform crossover is used which exchanges instructions between two individuals.

2.2 Cartesian genetic programming

Cartesian Genetic Programming was originally developed by Miller et al. [27] for the purpose of evolving digital circuits. However, the term ‘Cartesian Genetic Programming’ first appeared in a paper by Miller in 1999 [23] and was proposed as a general form of Genetic Programming in 2000 [26].

CGP represents a program as a directed graph (that for feed-forward functions is acyclic). The benefit of this type of representation is that it allows the implicit re-use of nodes, as a node can be connected to the output of any previous node in the graph, thereby allowing the repeated re-use of sub-graphs. This is an advantage over tree-based GP representations (without ADFs) where identical sub-trees have to be constructed independently.

In CGP, the genotype is a fixed length representation consisting of a list of integers which encode the function and connections of each node in the directed graph. However, CGP uses a genotype-phenotype mapping that does not require all of the nodes to be connected to each other, which results in the program (phenotype) being bounded but having variable length. Thus there may be genes that are entirely inactive, having no influence on the phenotype, and hence on the fitness. Such inactive genes therefore have a neutral effect on genotype fitness. This phenomenon is often referred to as neutrality. The influence of neutrality in CGP has been investigated in detail [25, 26, 39, 48, 49] and has been shown to be extremely beneficial to the efficiency of the evolutionary process on a range of test problems.

In CGP, each node in the directed graph represents a particular function and is encoded by a number of genes. One gene encodes the function that the node represents, and the remaining genes encode where in the graph the node takes its inputs from. The nodes take their inputs in a feed-forward manner from either the output of nodes in a previous column or from a program input (terminal). Also, the number of inputs that a node has, is dictated by the arity of the function it represents. The program data inputs are given the data addresses 0 to $n - 1$ where n is the number of program inputs. The data outputs of nodes in the genotype are given sequentially addresses, column by column, starting from n to $n + m - 1$ where m is the user-determined upper bound of the number of nodes (equal to the number of rows multiplied by the number of columns). If the problem requires k program outputs, then k integers are added to the end of the genotype, each one being the address of the output of a node where the program output is taken from. The two dimensional general form of a Cartesian Genetic Program is shown in Fig. 1.

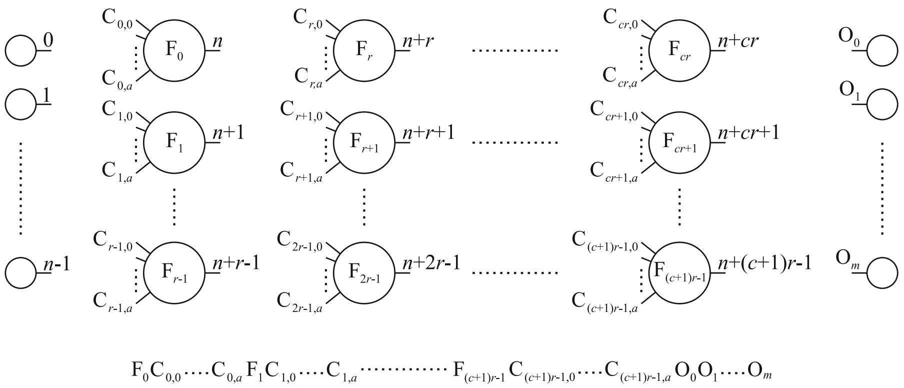
Figure 2 shows a CGP genotype and the corresponding phenotype that arose in the evolution of a 2-bit parallel multiplier (one row was used in this case). Figure 3 shows how the CGP genotype is decoded to produce a phenotype.

In general, CGP uses a $(\mu + \lambda)$ *evolutionary strategy* [34] with $\mu = 1$ and $\lambda = 4$, giving a population size of 5. The μ value indicates the number of individuals promoted to the next generation as parents and the λ value indicates the number of offspring generated from the promoted parents. The (1 + 4) evolutionary strategy is defined in Algorithm 1.

Algorithm 1: The (1 + 4) evolutionary strategy

```
1: for all $i$ such that $0 \leq i < 5$ do
2:     Randomly generate individual $i$
3: end for
4: Select the fittest individual, which is promoted as the parent
5: while a solution is not found and the generation limit is not reached do
6:     for all $i$ such that $0 \leq i < 4$ do
7:         Mutate the parent to generate offspring $i$
8:     end for
9:     Select the fittest individual using the following rules:
10:    if a single offspring has a better or equal fitness than the parent then
11:        The best offspring is promoted as the new parent
12:    else if many offspring have an equal fitness which is a better or equal fitness than the parent then
13:        A randomly selected offspring is promoted as the new parent
14:    else
15:        The parent is promoted
16:    end if
17: end while
```

In the rules for selecting the fittest individual (on lines 10–15 of Algorithm 1), an offspring is always chosen over the parent when they both have equal fitness, as the offspring is phenotypically identical (in terms of fitness) but genetically different to the parent. This allows neutral exploration of the search space until a phenotypically



0
1
n-1
F0
n
F1
n+1
Fr-1
n+r-1
Fr
n+r
Fr+1
n+r+1
F2r-1
n+2r-1
Fcr
n+cr
Fcr+1
n+cr+1
F(c+1)r-1
n+(c+1)r-1
O0
O1
Om
F0C0,0....C0,a F1C1,0....C1,a .... F(c+1)r-1 C(c+1)r-1,0....C(c+1)r-1,a O0O1....Om

Fig. 1 General form of two-dimensional CGP. It is a grid of nodes whose functions are chosen from a set of primitive functions. Two parameters *c*, and *r*, respectively, define the number of columns and rows in the grid. Each node is assumed to take as many inputs as the maximum function arity a. Every data input and node output are labelled consecutively (starting at 0) which gives it a unique data address which specifies where the input data or node output value can be accessed (shown in the figure on outputs of inputs and nodes). Nodes in the same column cannot be connected to each other. In most cases the graph is directed (as in this paper) so that a node may only have its inputs connected to either input data or the output of a node in a previous column. In general there may be a number of output genes (O_i) which specify where the program outputs are taken from. The structure of the genotype is seen below the schematic. All node function genes f_i are integer addresses in a look-up table of functions. All connection genes C_{ij} are data addresses and are integers taking values between 0 and the address of the node at the bottom of the previous column of nodes

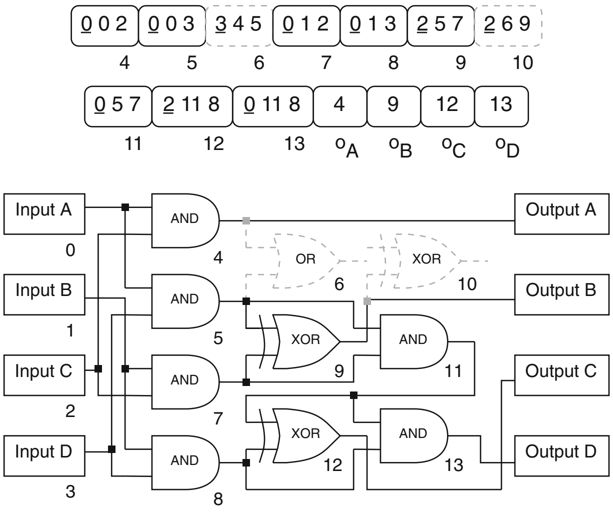


Fig. 2 A CGP genotype and corresponding phenotype for a 2-bit multiplier circuit. The underlined genes in the genotype encode the function of each node. The function look-up table is: AND(0), AND with one input inverted(1), XOR(2) and OR(3). The addresses are shown underneath each program input and node in the genotype and phenotype. The inactive areas of the genotype and phenotype are shown in *gray dashes* (nodes 6 and 10)

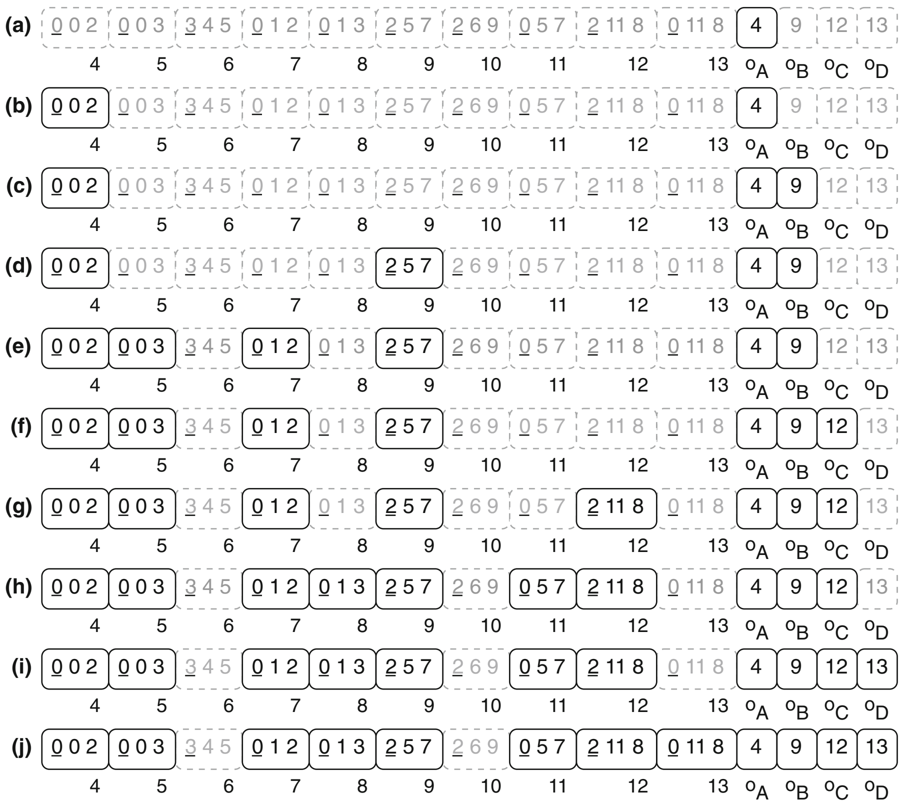


Fig. 3 The decoding procedure of a CGP genotype for the 2-bit multiplier problem. *a* Output A (o_A) connects to the output of node 4, move to node 4. *b* Node 4 connects to the program inputs 0 and 2, therefore the output A is decoded. Move to output B. *c* Output B (o_B) connects to the output of node 9, move to node 9. *d* Node 9 connects to the output of nodes 5 and 7, move to nodes 5 and 7. *e* Nodes 5 and 7 connect to the program inputs 0, 3, 1 and 2, therefore output B is decoded. Move to output C. The procedure continues until output C (o_C) and output D (o_D) are decoded (steps *f*) to *h* and steps *i* to *j*, respectively). When all outputs are decoded, the genotype is fully decoded

better offspring is discovered. If multiple offspring are tied for fitness (lines 12–13 of Algorithm 1), then one of the offspring is chosen at random to be promoted.

The mutation operator used in CGP is typically a point-mutation operator, in which a number of randomly chosen genes in the genotype are changed to other valid randomly chosen values. If a function gene is chosen for mutation, then a valid value would be the address of any function in the function set. Whereas if an input gene is chosen for mutation, then a valid value would be the address of the output of any previous node in the genotype or of any program input. Also, a valid value for a program output gene is the address of the output of any node in the genotype but not the address of a program input. The number of genes in the genotype that can be mutated in a single application of the mutation operator is defined by the user, and is normally a percentage of the total number of genes in the genotype or chromosome. An example of the point mutation operator is shown

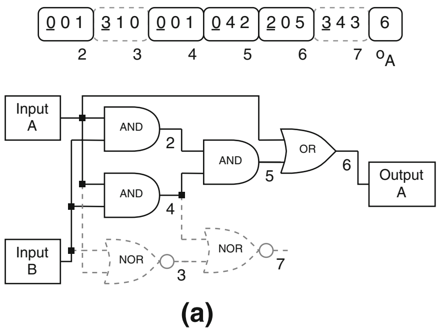


Fig. 4 An example of the point-mutation operator before and after it is applied to a CGP genotype and the corresponding phenotypes. A single point mutation occurs in the program output gene (o_A), changing the value from 6 to 7. This causes nodes 3 and 7 to become active, whilst making nodes 2, 5 and 6 inactive. The inactive areas are shown in *gray dashes*

in Fig. 4, which highlights how a small change in the genotype can produce an enormous change in the phenotype.

A crossover operator is not used in single-chromosome CGP, as a suitable operator is yet to be discovered. Originally, a 1-point crossover operator was used in CGP (similar to the *n*-point crossover in GAs) but was found to be disruptive to the sub-graphs within the chromosome, and had a detrimental affect on the performance of CGP [23]. Recent work by Clegg et al. [8] has investigated crossover in CGP (and GP in general). Their approach uses a floating-point crossover operator, similar to that found in Evolutionary Programming (EP) [11, 12], and also adds an extra layer of encoding to the genotype, in which all genes are encoded as a floating-point number in the range [0,1]. A larger population and tournament selection were also used instead of the (1 + 4) evolutionary strategy normally used in CGP, to try and improve the population diversity. The results of the new approach appear promising when applied to two symbolic regression problems, but further work is required on a range of problems in order to assess its advantages [8].

3 Multi-chromosome representations

3.1 Digital circuits

The difference between a CGP genotype (described earlier in Sect. 2.2) and a Multi-chromosome CGP genotype is that the Multi-chromosome CGP genotype is divided into a number of equal length sections called chromosomes. In the case of digital circuit evolution the number of chromosomes present in the genotype of an individual is dictated by the number of program outputs required by the problem, as each chromosome is connected to a *single* program output. This allows large problems with multiple outputs (normally encoded in a single genotype) to be broken down into many smaller problems (each encoded by a chromosome) with a

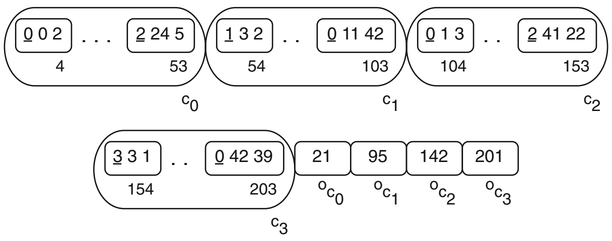


Fig. 5 A multi-chromosome CGP genotype encoding a 2-bit multiplier (four outputs, $o_{c0} - o_{c3}$) containing four chromosomes ($c_0 - c_3$), each consisting of 50 nodes

single output. For multiple output digital circuits the separate outputs are defined by its specification. In the case of image processing the outputs are defined by the number of parts the image is chosen to be divided into. The idea is that this approach should make the problem easier to solve. In the case of digital circuits, Torresen has shown that evolving the outputs to a problem incrementally improves evolvability [37]. However, in this paper all of the outputs are evolved simultaneously. We argue that by allowing each of the smaller problems to be encoded in a chromosome, the whole problem is still encoded in a single genotype but the interconnectivity between the smaller problems (which can cause obstacles in the fitness landscape) has been removed.

Each chromosome contains an equal number of nodes, and is treated as a genotype of an individual with a single program output. The inputs of each node encoded in a chromosome are only allowed to connect to the output of earlier nodes encoded in the same chromosome or any program input (terminals). This creates a form of compartmentalisation in the genotype which supports the idea of removing the interconnectivity between the smaller problems encoded in each chromosome. An example of a Multi-chromosome CGP genotype for a digital circuit problem (2-bit multiplier) is shown in Fig. 5. The 2-bit multiplier problem has four outputs, so it is broken down into four smaller problems. Each of the smaller problems has one output and is encoded in a single chromosome.

The multi-chromosome approach to CGP shares some similarities with another GP technique known as Parisian GP [9], which is inspired by the Michigan Approach to Classifier Systems, in that both techniques form a solution to a problem from sub-solutions. However, in Parisian GP, an individual only represents part of a solution, and the whole solution is made up of a set of individuals from the population. This differs from the multi-chromosome approach to CGP, as each chromosome encodes a solution to a distinct sub-problem and the solution to the entire problem is contained in a single individual (genotype), which consists of a number of chromosomes, each encoding a different sub-problem. Another difference between the two techniques is that Parisian GP uses two separate fitness functions; a local fitness function to assess each individuals contribution and a global fitness function to evaluate how well the set of individuals solves the problem. Whereas the multi-chromosome approach to CGP uses a single fitness function to evaluate how well each chromosome solves the sub-problem it has been assigned. When all of the sub-problems are solved, the entire problem is also solved.

3.2 Multi-chromosome evolutionary strategy

Rather than assigning a single fitness value to a number of program outputs, as in the single chromosome CGP, a fitness value is assigned to the output of each chromosome in Multi-chromosome CGP, as each chromosome's output is also a program output. Therefore, if an individual with a multi-chromosome genotype has *n* program outputs, the individual's genotype contains *n* chromosomes, and the individual has *n* fitness values. This allows each chromosome of an individual to be compared with the corresponding chromosome of other individuals in the population, by using a variation of the (1 + 4) evolutionary strategy (described earlier in Sect. 2.2) called the (1 + 4) *multi-chromosome evolutionary strategy*.

The (1 + 4) multi-chromosome evolutionary strategy selects the best chromosome at each position from all of the individuals in the population and generates a new best of generation individual containing the fittest chromosome at each position. The new best of generation individual may not have existed in the population, as it is a combination of the best chromosomes from all the individuals, so it could be thought of as a "super" individual. The multi-chromosome version of the (1 + 4) evolutionary strategy therefore behaves as an intelligent multi-chromosome crossover operator, as it selects the best parts from all the individuals. The overall fitness of the new individual (i.e. the sum of the fitness scores for all chromosomes of an individual) will also be better than or equal to the fitness of any individual in the population from which it was generated. An example of the

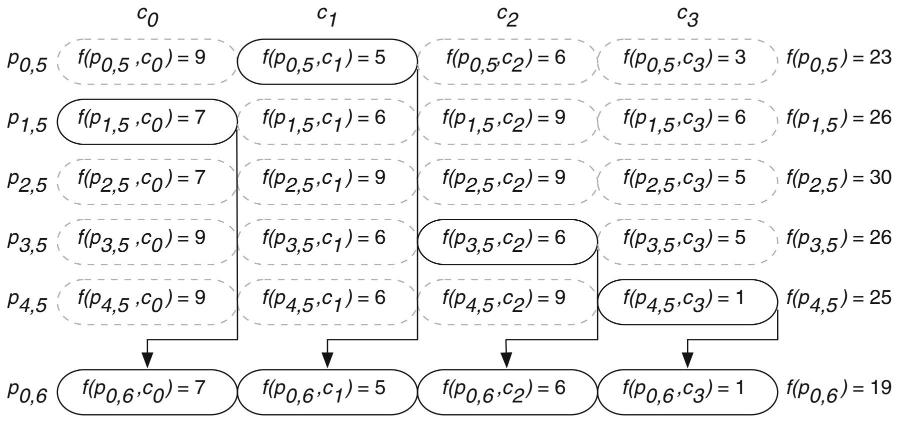


Fig. 6 The (1 + 4) multi-chromosome evolutionary strategy used in Multi-chromosome CGP. $p_{x,g}$—parent x at generation g, c_y—chromosome y, $f(p_{x,g}, c_y)$—fitness of chromosome y in parent x at generation g, $f(p_{x,g})$—fitness of parent x at generation g

multi-chromosome evolutionary strategy is shown in Fig. 6, whilst an outline of the (1 + 4) multi-chromosome evolutionary strategy is shown in Algorithm 2.

Algorithm 2: The (1 + 4) multi-chromosome evolutionary strategy

```
1: for all $i$ such that $0 \le i < 5$ do
2:     Randomly generate individual $i$
3: end for
4: Select the fittest individual, which is promoted as the parent
5: while a solution is not found and the generation limit is not reached do
6:     for all $i$ such that $0 \le i < 4$ do
7:         Mutate the parent to generate offspring $i$
8:     end for
9:     Generate the fittest individual using the following rules:
10:    for all $j$ such that $0 \le j <$ number of chromosomes do
11:        if an offspring chromosome $j$ has a better or equal fitness than the parent chromosome $j$ then
12:            Offspring chromosome $j$ is promoted as the new parent chromosome $j$
13:        else if many offspring chromosome $j$'s have an equal fitness which is a better or equal fitness
               than the parent chromosome $j$ then
14:            A randomly selected offspring chromosome $j$ is promoted as the new parent chromosome $j$
15:        else
16:            The parent chromosome $j$ is promoted
17:        end if
18:    end for
19: end while
```

3.3 Mammography

Mammograms are high resolution X-ray images of the breast which are used as a diagnostic as well as a screening tool for breast cancer. The process of identifying and evaluating signs of cancer from mammograms is a very difficult and time-consuming task that requires skilled and experienced radiologists. This assessment is also, by its nature, highly subjective and susceptible to error, leading to cancers being missed and the patients misdiagnosed. To achieve a more accurate and reliable diagnosis, Computer Aided Detection (CAD) systems have been investigated which provide an objective, quantitative evaluation. CAD systems have the potential to help in two main ways: (i) the detection of suspicious areas in the mammogram that require further investigation and (ii) the classification of such areas as cancerous (malignant) or non-cancerous (benign) [13].

The long term aim of this work is to assess the potential benefit of a new representation of evolutionary algorithm in the classification of mammograms as part of a CAD system and determine whether further development of such algorithms will lead to a more confident diagnosis. One important feature of mammograms are *microcalcifications*—small deposits of calcium whose size, shape and distribution within the breast tissue have been used to help diagnosis of cancer.

In this paper, we investigate the application of Multi-chromosome CGP to the characterisation and classification of microcalcifications. Selected regions of interest (RoI) of mammograms from the Lawrence Livermore National Laboratory database [1] have been cropped to a uniform size of 128 $\times$ 128 pixels, each providing at least one microcalcification for classification. The size restriction of 128 $\times$ 128 pixels is faster to process than larger images and has previously been used by [16]. Furthermore, it is common to only use a RoI for classification purposes. RoIs containing microcalcifications are typically found by a detection system, such as that described in [18] or selected manually, as described in [40].

The images are then divided into 256, non-overlapping 8 $\times$ 8 pixel areas. The 64 grey scale pixel values (0–255) for each of these areas form the inputs to an individual chromosome. Thus, each genotype consists of 256 independent CGP chromosomes, each of which uses a large grid of 32 rows and 128 columns. This provides a level of redundancy proven to be advantageous, as reported in previous work by Miller and Smith [25]. Each chromosome has a single output gene.

The fitness of chromosome c_i is defined using Eqs. 1 and 2. Firstly, the output of the chromosome, o_i (a value between [0,255]), is rescaled ($r(o_i)$) into the range [0,127] or [128,255] depending on whether it is classified as malignant or benign by the output threshold, oth (also a value in the range [0,255]). The fitness of the chromosome, $f(c_i)$, is then determined by whether the radiologist classifies the section of the image corresponding to the chromosome as malignant or benign.

$$r(o_i) = \begin{cases} 128 \times \frac{o_i}{o\text{th}} & \text{if } o_i < o\text{th}, \\ 128 \times \frac{o_i - o\text{th}}{255 - o\text{th}} + 128 & \text{if } o_i \geq o\text{th}. \end{cases} \tag{1}$$

$$f(c_i) = \begin{cases} 255 - r(o_i) & \text{if image section classified malignant,} \\ r(o_i) & \text{if image section classified benign.} \end{cases} \tag{2}$$

For the mammography application, a (1 + 2) multi-chromosome evolutionary strategy was used. A chromosome rearrangement stage is then applied to the "super" individual, as described in Algorithm 3. According to a re-arrangement mutation rate, the chromosomes may either be swapped or replaced with another of the 256 chromosomes, chosen at random. A number of rearrangement strategies are investigated over independent evolutionary runs, which consist of:

1. a *random swap*, in which any chromosome might be swapped with another.
2. a *neighbouring swap*, in which a part might only be swapped at random with its four direct spatial neighbours. The neighbouring swap has been implemented to target structures that continue from one part of the image to the next. Neighbouring parts might therefore have similar image properties and are likely to respond equally well to the same chromosome.
3. a *copying operation*, where a random chromosome is chosen to overwrite a different chromosome (re-use).

Algorithm 3: The chromosome rearrangement strategy

```
1: The “super” individual is constructed according to Algorithm 2
2: for all $i$ such that $0 \leq i <$ number of offspring do
3:     Generate offspring by point mutation of the “super” individual
4:     Apply rearrangement mutation to the offspring
5: end for
6: Offspring and “super” individual form next generation
```

If after a mutational rearrangement the fitness of an individual’s chromosome declines compared to its fitness before the swap, then the rearrangement is disallowed. However, if the rearrangement makes an improvement to the resulting fitness, then the exchange is preserved. Although there is a risk that the diversity of chromosomes might be reduced by deleting ones that do not perform well and substituting them with a copy of a fitter one, this approach gives the genotypes a higher opportunity for individual mutation which in itself has the potential of restoring diversity to some extent. We can see this because if every chromosome is unique (no copies) then mutations can only be beneficial independently. If there are duplicated chromosomes, any mutation occurring in those would have to be, on average, beneficial to all of them. This means one chromosome’s fitness might be reduced if all other copies gained a higher fitness, through the rearrangement.

4 Experiment details

4.1 Computational effort

In each experiment on digital circuits, the results for all independent runs were assessed using a statistic called *computational effort*. This metric was introduced by Koza in [17], as a measure of the computational effort required to solve a problem based on the data from all of the independent runs. The formula to calculate computational effort is shown in Eq. 3. The notation is taken from [17] as follows: $N_s(i)$—the number of successful independent runs by generation i, N_{total}—the total numberof independent runs, $P(M, i)$—the cumulative probability ofsuccess for an independent run with population size M producing a solution by generation i, $R(P(M, i), z)$—the number of independent runs required to satisfy the success predicate by generation i with probability z, $I(M, i, z)$—the number of individuals that need to be processed to produce a solution with probability z, using population size M, at generation i, CE—the minimum number of individuals to be processed with probability z, using population size M, hence the minimum computational effort. In this paper, we use $z = 0.99$.

$$\begin{aligned} P(M,i) &= \frac{N_s(i)}{N_{\text{total}}} \\ R(P(M,i),z) &= \left\lceil \frac{\log(1-z)}{\log(1-P(M,i))} \right\rceil \\ I(M,i,z) &= M \times R(P(M,i),z) \times (i+1) \\ CE &= \min_i I(M,i,z) \end{aligned} \tag{3}$$

The computational effort statistic used is a popular performance measure in the GP community. However, it is by no means perfect and has numerous inadequacies. Christensen and Oppacher [7] found that the *ceiling* operator in Eq. 3 has a tendency to overestimate $R(z)$, whilst the *min* operator tends to underestimate the computational effort required. Furthermore, the underestimation increases in systems with a high number of generations, which is the case in the approach used in this paper. Niehaus and Banzhaf [28] later found that the underestimation of the computational effort statistic is inversely proportional to the number of runs used in the calculation, so for a small number of runs, the underestimation of computational effort is very large. In this paper, only 50 independent runs are used (which is classed as a small number of runs) for each experiment, as this was the number of runs used in the work we compare with. Therefore, the computational effort figures are likely to be underestimates of the theoretical value for computational effort and should only be used as a rough guide. However, Niehaus and Banzhaf [28] also found that as the probability of a run ending in failure increased, the computational effort deviated further from the theoretical value. In Sect. 5, every run continues until a solution is found, thereby producing a 100% success rate, which should improve the accuracy of the computational effort values.

4.2 Confidence interval

Walker et al. [45, 46] state that computational effort is only a point statistic, with no confidence interval, so any comparisons made with other techniques are inconclusive. However, they do propose an approach for defining a 95% confidence interval for the true computational effort of a technique, using Wilson's method. The approach starts by defining formulae for the upper and lower bounds of the confidence interval for the number of successful runs given a probability z, which are shown in Eqs. 4 and 5. The proportion of successes, p, is defined as $p = r/n$, where r is the number of successful runs, and n is the total number of runs. The z_{norm} value is set to 1.96, as this was used in [45, 46].

$$upper(p,n) = \frac{2np + z_{\text{norm}}^2 + z_{\text{norm}}\sqrt{z_{\text{norm}}^2 + 4np(1-p)}}{2(n + z_{\text{norm}}^2)} \tag{4}$$

$$lower(p,n) = \frac{2np + z_{\text{norm}}^2 - z_{\text{norm}}\sqrt{z_{\text{norm}}^2 + 4np(1-p)}}{2(n + z_{\text{norm}}^2)} \tag{5}$$

Equations 4 and 5 can then be used to define the upper and lower bounds of the confidence interval for the true computational effort at generation i, $\tau(i)$, by substituting *upper*(p, n) and *lower*(p, n) for $P(M, i)$ in Eq. 3 and dropping the ceiling operator from the $R(P(M, i), z)$ formula, as shown in Eq. 6.

$$MR(upper(p,n),z)(i+1) \leq \tau(i) \leq MR(lower(p,n),z)(i+1) \qquad (6)$$

The confidence interval produced by Eq. 6 is always valid regardless of the probability of success or the number of runs [45, 46].

In order to find the confidence interval for the true minimum computational effort, $\tau(j)$, the minimum generation, j, and the proportion of successes, p, at which the minimum computational effort occurs must be known. However, this is virtually always unknown when using any form of evolutionary computation. Therefore, a good estimate for the minimum computational effort is required, in order to find these values. Koza's proposed approach for calculating the minimum computational effort (Eq. 3) does provide a good estimate for the minimum generation, and also the proportion for successes. Using this estimate for the minimum generation and proportion of successes makes it possible to calculate the confidence interval for the estimated minimum computational effort [45, 46].

4.3 Non-parametric statistics

Since there are still questions concerning the accuracy of the computational effort statistic, a variety of other statistics have also been compiled. The results from the digital circuit experiments in Sect. 5 are positively skewed (i.e. not normally distributed), since the minimum number of evaluations is 1. Therefore, parametric measures, such as the mean and standard deviation can not used, as they would not provide an accurate and meaningful representation of the data. Hence, a number of non-parametric statistics are used. These are the median number of evaluations, median absolute deviation (MAD), and inter-quartile range (IQR). The MAD is a measure of variability within a distribution, and is similar to standard deviation, except it is based on the median rather than the mean. Also, the IQR measures the dispersion of the middle 50% of the distribution, and is the difference between the third and first quartiles.

The significance of the results for Sect. 5 have also been assessed using the non-parametric Mann-Whitney U test [21] (also known as the Wilcoxon Rank-sum test [47]), which assesses whether two independent samples come from the same distribution.

In order to allow authors to compare with the figures presented in this paper and conduct their own statistical tests, the CGP data sets collected from all runs will be made available from the CGP website.[1]

5 Digital circuit results

The performance of both the multi-chromosome and single chromosome versions of CGP were tested on a number of multiple output digital circuit problems shown in Table 1 with their corresponding number of inputs and outputs.

[1] The CGP website is currently under construction and can be found at http://www.cartesiangp.co.uk.

Table 1 The digital circuit problems used to test the performance of the single and multi-chromosome versions of CGP

Digital circuit	Number of inputs	Number of outputs
2-bit Adder (Add)	5	3
3-bit Adder (Add)	7	4
2-bit Multiplier (Mul)	4	4
3-bit Multiplier (Mul)	6	6
3:8-bit De-multiplexer (DeMUX)	3	8
4 × 1-bit Comparator (Comp)	4	18
3-bit Arithmetic Logic Unit (ALU)	8	17

The abbreviation for each problem is shown in parenthesis

The n-bit adder takes two n-bit integers and a 1-bit carry-in ($2n + 1$ inputs) and performs addition to produce a n-bit integer output and a 1-bit carry-out ($n + 1$ outputs). The n-bit multiplier takes two n-bit integers ($2n$ inputs) and multiplies them together to produce a $2n$-bit integer ($2n$ outputs). The 3:8-bit de-multiplexer converts a signal consisting of three components (3 inputs), which has already been compressed by a multiplexer, back into its original uncompressed signal consisting of eight components (8 outputs). The 4 × 1-bit comparator takes four 1-bit integers (4 inputs) and compares every possible pair combination of them to find out if the first number of the pair is less than, equal to or greater than the second number of the pair (six pair combinations each with 3 outputs, totalling 18 outputs overall). An example is shown in Fig. 7a to illustrate the point. The final problem tested was a possible implementation of a 3-bit arithmetic logic unit (ALU), which takes two 3-bit integers and a low and a high carry-in (totalling 8 inputs) and performs the functions of addition, subtraction, multiplication and protected division, all in parallel, on the two 3-bit integers to produce two 4-bit numbers from addition and subtraction, a 6-bit number from multiplication and a 3-bit number from protected division (17 outputs overall). This is illustrated in Fig. 7b.

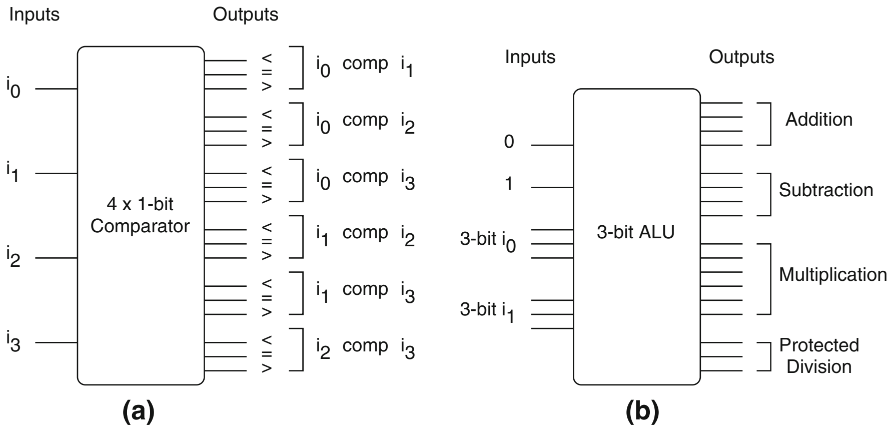


Fig. 7 Examples of the 4 × 1-bit comparator (**a**) and the 3-bit ALU (**b**) showing the inputs and outputs of each circuit. In **a**, each of the six output blocks constitutes a comparison, and produces a result of less than (<), equal to (=) or greater than (>) at one of the three outputs for that block

Table 2 The parameters used for CGP

Parameter	Value
Population size	5
Initial chromosome size (nodes/genes)	100/300
Initial genotype size	Initial chromosome size × number of chromosomes
Genotype point mutation rate (%)	3
Genotype point mutation probability	1
Number of independent runs	50

The parameters used for the multi-chromosome and single chromosome versions of CGP are shown in Table 2. The operator rates and probabilities were determined to be fairly optimal by means of an experimental optimisation process in previous work [41–43]. The digital multiplier and Arithmetic Logic Unit problems used a function set consisting of: AND, AND (one input inverted), OR and XOR, whilst the other problems used the function set: AND, NAND, OR and NOR.

In most GP experiments, the maximum number of generations allowed is set quite low (for example, 1,000 generations), normally resulting in a success rate that is less than 100%. However, the aim of each experiment in this section is to achieve 100% success. Therefore, the maximum number of generations allowed for each independent run was set to 20 million. This allows ample time for each independent run to find a solution (at which point the independent run terminates), whilst guaranteeing that each independent run always terminates after 20 million generations, if a solution is not found.

5.1 Results and discussion

The method used for comparing the techniques is the computational effort statistic described in Sect. 4.1. The computational effort figures for both the multi-chromosome and single chromosome versions of CGP are calculated over fifty independent runs and are shown in Table 3. Unfortunately, GP researchers tend to avoid problems with multiple outputs, therefore we have no figures for GP (with and without ADFs) to perform fair comparisons with multi-chromosome CGP. However, Walker et al. have shown that CGP performs favourably when compared with GP (with and without ADFs) on a number of problems (including some single output digital circuit problems) [44].

For all of the digital circuit problems tested, CGP and Multi-chromosome CGP produced 100% successful solutions, except for the 3-bit Arithmetic Logic Unit, where CGP failed to find a solution after twenty million generations. This gives a good indication of how difficult it is to evolve a solution to the 3-bit Arithmetic Logic Unit problem.

Comparing the results for CGP and Multi-chromosome CGP, it is clear that the use of multiple chromosomes to break down the test problems into smaller, simpler problems provides a distinct advantage. Multi-chromosome CGP significantly

Table 3 The computational effort (CE) figures for CGP and Multi-chromosome CGP (MC-CGP)

	CGP			MC-CGP		
Problem	CI_{lower}	CE	CI_{upper}	CI_{lower}	CE	CI_{upper}
2-bit Add	577,067	834,246	1,237,329	82,718	140,800	168,025
3-bit Add	6,315,127	8,599,682	12,827,903	700,668	1,286,000	1,442,657
2-bit Mul	24,675	33,602	50,123	5,971	11,200	12,362
3-bit Mul	16,448,737	24,152,005	33,867,501	513,220	873,600	1,042,503
3:8-bit DeMUX	44,180	75,000	89,742	2,346	4,400	4,858
4 × 1-bit Comp	2,304,080	3,922,000	4,680,272	5,876	10,000	11,936
3-bit ALU	–	–	–	1,016,914	1,908,000	2,105,517

Also included are the lower (CI_{lower}) upper and (CI_{upper}) bounds of the confidence interval for the true computational effort

Table 4 The median number of evaluations (ME), median absolute deviation (MAD), and inter-quartile range (IQR) of CGP and Multi-chromosome CGP (MC-CGP)

	CGP			MC-CGP			
Problem	ME	MAD	IQR	ME	MAD	IQR	*U*
2-bit Add	132,565	76,228	178,335	29,239	13,350	25,663	$133^{\ddagger}$
3-bit Add	1,943,585	996,482	2,174,500	174,625	82,050	222,178	$96^{\ddagger}$
2-bit Mul	6,197	4,130	8,489	1,721	662	1,872	$315^{\ddagger}$
3-bit Mul	4,030,201	2,181,656	6,110,863	140,643	52,722	102,960	$0^{\ddagger}$
3:8-bit DeMUX	13,797	4,842	9,635	845	274	524	$0^{\ddagger}$
4 × 1-bit Comp	770,475	335,916	672,160	1,693	704	1,442	$0^{\ddagger}$
3-bit ALU	–	–	–	304,345	127,312	246,988	–

Also shown are the *U* values from the Mann-Whitney significance test when comparing CGP and Multi-chromosome CGP (MC-CGP)

‡ The *U* values are classed as highly significant ($P < 0.001$)

outperforms CGP on all of the problems tested. Multi-chromosome CGP approximately improves performance between 3 and 392 times when compared with CGP (see Table 3). It is also worth noting that the speedup increases with problem complexity on the adder and multiplier problems, implying that Multi-chromosome CGP may perform even better on larger, more complex problems of this nature. These results are supported by the statistics and the results of the Mann-Whitney significance test in Table 4, as every comparison between the single and multi-chromosome approaches of CGP are classed as highly significant, indicating that there is only a very small probability ($P < 0.001$) that the results of the single and multi-chromosome techniques are from the same distribution.

The variance between the speedup for Multi-chromosome CGP compared with CGP for different problems, appears to be related to the number of problem outputs. Notice how the biggest speedup recorded was found on the 4 × 1-bit comparator problem, which is also the problem with the most outputs. This suggests problem

Table 5 The computational effort (CE) figures for CGP with the same initial genotype length (*len*), in terms of nodes, as the multi-chromosome approach

		CGP		
Problem	*len*	CI_{lower}	CE	CI_{upper}
2-bit Add	300	275,645	469,200	559,917
3-bit Add	400	4,811,660	8,190,400	9,773,914
2-bit Mul	400	30,550	52,000	62,057
3-bit Mul	600	35,231,021	65,416,400	78,074,575

Also included are the lower (CI_{lower}) and upper (CI_{upper}) bounds of the confidence interval for the true computational effort

complexity and the number of program outputs are directly linked, implying the multi-chromosome approach is less affected by an increase in problem complexity than the single chromosome approach.

To rule out the possibility that the increased overall genotype length in the multi-chromosome approach is responsible for the performance difference between the single and multi-chromosome versions of CGP, further runs of CGP were performed on the adder and multiplier problems, where the initial genotype length is equivalent to that of the multi-chromo-some approach. The computational effort figures from these runs are shown in Table 5. By comparing the figures in Table 5 with the previous results for CGP in Table 3, it is possible to see that giving CGP extra resources by increasing the genotype length does improve the performance of the techniques on the adder problem, but is actually detrimental to the performance on the multiplier problem. However, the performance on the adder problem is still much worse than that of Multi-chromosome CGP. This provides further evidence that breaking down a complex, difficult problem into many smaller, simpler problems that are co-evolved is beneficial to performance.

The noticeable speedup caused by the use of multiple chromosomes in CGP clearly indicates that the approach could be used to evolve much harder, multiple-output problems (such as digital circuits), which CGP currently fails to solve. The only drawback of the multi-chromosome approach is the solutions are much larger (in terms of number of nodes used) than the optimal solution. However, our objective in this paper is not to find efficient solutions, our main concern is with improving performance.

The larger solutions appear to be a result of severing the interconnections between the smaller problems, as early sections of the evolved solution which are normally re-used later in the solution are being replicated. However, one advantage of the representation used in the Multi-chromosome CGP is the whole solution is present in a single genotype, which can be easily converted to a single chromosome genotype (as in CGP), by removing the chromosome restrictions. This feature of the representation allows us in our future work to investigate the use of Multi-chromosome CGP to evolve efficient solutions, using the technique from [24] to reduce the size of the evolved solutions. Once a solution is found, the fitness function is changed to minimise the total number of nodes used in the phenotype of

Table 6 The computational effort (CE) figures for Multi-chromosome CGP (MC-CGP$^{\diamond}$) with a (1 + 4) evolutionary strategy

	MC-CGP$^{\diamond}$		
Adder	$\mathrm{CI}_{\mathrm{lower}}$	CE	$\mathrm{CI}_{\mathrm{upper}}$
2-bit	140,262	249,600	276,764
3-bit	3,938,076	7,008,000	7,770,571

Also included are the lower ($\mathrm{CI}_{\mathrm{lower}}$) and upper ($\mathrm{CI}_{\mathrm{upper}}$) bounds of the confidence interval for the true computational effort

the solution (either by minimising each chromosome phenotype or the phenotype of the whole solution). This allows the evolved solutions from Multi-chromosome CGP to be minimised to a size comparable to those found using CGP.

5.2 Investigating the multi-chromosome evolutionary strategy

To see how much of an impact the multi-chromosome evolutionary strategy had on the results, further experiments were carried out on the 2-bit and 3-bit adder problems using Multi-chromosome CGP with the (1 + 4) evolutionary strategy used in CGP, instead of the (1 + 4) multi-chromosome evolutionary strategy. This has the effect of grouping all of the chromosomes within a genotype together, and treating the individual like a CGP genotype that has been compartmentalised. Therefore, good chromosomes are not allowed to be exchanged between individuals in the selection process to form and promote a “super” individual.

Comparing the results from Tables 3 and 6 clearly shows the use of a (1 + 4) evolutionary strategy with Multi-chromosome CGP does not perform as well as Multi-chromosome CGP with the (1 + 4) multi-chromosome evolutionary strategy. This implies the use of the multi-chromosome evolutionary strategy to select the fittest individual in the population (by selecting the fittest chromosomes from each position) is beneficial to the performance of Multi-chromosome CGP, in contrast to the selection of the fittest individual based on the individual’s overall fitness (the sum of all of its chromosome fitness values). However, Multi-chromosome CGP with a (1 + 4) evolutionary strategy does perform marginally better than the single chromosome version of CGP with a (1 + 4) evolutionary strategy, respectively. Therefore implying the multi-chromosome evolutionary strategy is not solely responsible for the improvement in performance, but the use of a multi-chromosome representation, as opposed to a single chromosome representation (as in CGP), also improves the performance of CGP.

6 Mammography results

The chromosome mutation rate defines the percentage of genes in each chromosome that are mutated. The function set is shown in Table 7, where *x* and *y* represents the

Table 7 Parameters for multiple CGP network

Parameter	Value
No. parts per image	256 (16 × 16)
Part size (Pixels)	8 × 8
Chromosome rearrangement rate (%)	3
Chromosome mutation rate (%)	1
No. runs	10
No. generations	1000
No. columns in each CGP network	128
No. rows in each CGP network	32
Function set	x, $x + y$, $\lvert x - y\rvert$, $\lvert 2x - y\rvert$, $x \,\&\, y$, $max(x, y)$, $min(x, y)$

bitwise AND function. The output from all node operations is kept within the range [0,255], by truncation.

Images used in this study are constructed from mammograms in the Lawrence Livermore National Laboratory database that feature microcalcifications [1]. In each case, a RoI consisting of a 128 × 128 pixel image is constructed containing at least one microcalcification from a particular mammogram. Each RoI image is divided into 256 parts and the status of each part labelled as either being benign or malignant according to the radiologist.

When an image is processed by the system the output value generated for each CGP chromosome is compared to a predetermined threshold. An output value below the threshold is interpreted as an indication of *malignancy* and an output value above the threshold is an indication of *benign* tissue. In this study, output values ranged from 0 to 255 and the threshold adopted was 4. This bias toward benign results reflects the relative scarcity of malignant areas within the images (some images having no malignant areas). As described in Sect. 6, a fitness value can then be calculated on the basis of this predicted value and the predetermined status of that part of the image as identified by the radiologist.

6.1 Training image classification

In total 31 images were created, of which 13 contained malignant microcalcifications and 18 benign microcalcifications. A random selection of 21 images (8 malignant, 13 benign) were used for training the Multi-chromosome CGP program, and the remaining 10 images (5 malignant, 5 benign) were used for the testing stage. Although the ratio of benign to malignant images may not, on the face of it represent the incidence of microcalcifications in clinical practice there are two factors that should be noted. Firstly, it is intended that this system be used to support existing clinical practice and, as such, the mammogram will be pre-screened by a radiologist or other software system to highlight "suspicious" areas. Secondly, each malignant image is split into parts, of which, typically, 95% represent tissue which is benign.

From the evolved individuals at the end of each of the 10 training runs, the best evolved chromosome from each chromosome position in the genotype was extracted to form a “super” individual (similar to the multi-chromosome evolutionary strategy) for the testing phase. The chromosomes in this “super” individual reached different fitness values, which on average range from 81.2% to 90.6% depending on the chromosome recombination method used. The results from the training stage are given in Table 8, which also details the performance of the chromosome rearrangement strategies. Graphs for average and best fitness are also given in Fig. 8.

Table 8 The best, average and worst fitness scores for the chromosomes in the CGP genotype after training

Recombination	Best (%)	Worst (%)	Average (%)
No swap, re-use	95.7	67.8	85.5
Neighbouring swap, no re-use	96.9	62.0	87.8
Neighbouring swap, re-use	96.9	63.9	89.0
Random swap, no re-use	96.5	70.6	89.4
Random swap, re-use	96.9	71.4	90.6

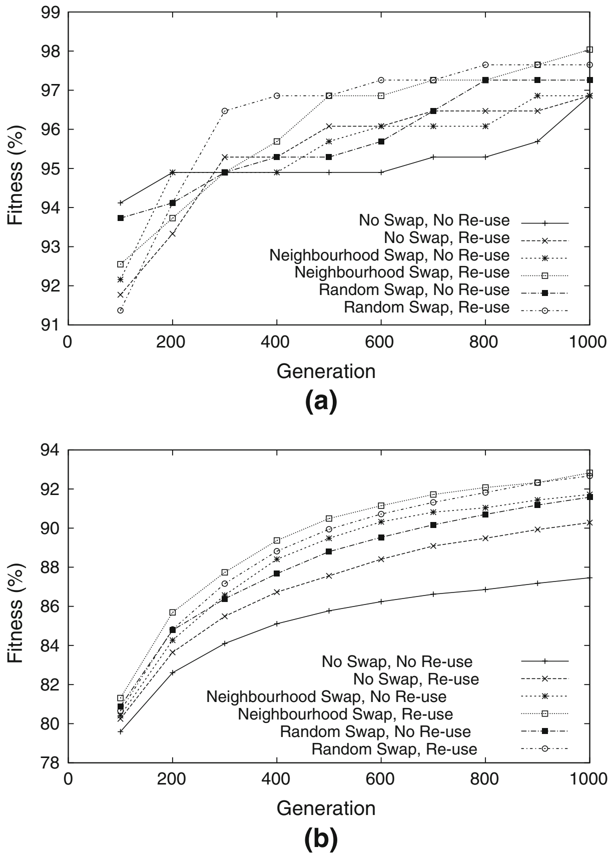


Fig. 8 The best (**a**) and average (**b**) fitness scores of the multi-chromosome CGP with different recombination strategies during the training phase

The results show that although all three recombination strategies (with and without re-use) perform well in terms of the best chromosome in the “super” individual, there are differences between how effective they are overall. It appears that the use of a chromosome swapping operator improves the worst and average chromosome fitness in the genotype. This could be attributed to the fact that the vast majority of the sections in training images are benign. If a chromosome that is a good benign classifier in one section of the image is able to swap with chromosomes in other benign sections of the image, then it can be trained on a larger and more diverse set of possible benign image sections. Likewise, it also allows the possibility for chromosomes that would not see a malignant section of the training images, under the no swap strategy, to actually gain experience of such sections.

It is also possible to see from the results that the ability to re-use chromosomes in other sections of the training images also improves the best, worst and average fitness of all three recombination strategies. This could also be attributed to the fact that the majority of the training images contain benign sections. If a chromosome is found which is a good benign classifier, it can quickly replicate to other chromosome locations that are classifying benign sections of the training images and thereby quickly improve the fitness score.

6.2 Test image classification

One of the problems that might occur when applying the evolved program to classifying the test images is that some of the CGP chromosomes may not have been trained on sections of images containing a microcalcification and therefore will only recognise background breast tissue. To overcome this problem every section of each test image is evaluated with every evolved CGP chromosome from the “super” individual from the training phase. Two possible classification approaches are used on the test images. The first investigates how well each single chromosome from the “super” individual can classify the entire test image. The second uses a voting procedure, in which a classification from every chromosome is made for each section of a test image. If the number of chromosomes that classify the section as malignant is greater than a user-defined threshold (V_{TH}), then the section of the image is classified as malignant.

The results from the first classification approach, which are based on the classification of all 256 single chromosomes are shown in Table 9. The classification breakdown for the best chromosome from each recombination technique is

Table 9 The best, average and worst test image classifications for each of the individual chromosomes

Recombination	Best (%)	Worst (%)	Average (%)
No swap, no re-use	80	20	53.3
No swap, re-use	80	30	51.4
Neighbouring swap, no re-use	80	20	50.3
Neighbouring swap, re-use	80	20	50.3
Random swap, no re-use	70	30	50.2
Random swap, re-use	70	50	50.2

Table 10 Test image classification by the best individual chromosomes

	Confusion matrices				Statistics					
Recombination	Bb	Bm	Mb	Mm	TP	TN	FP	FN	P	F_1
No swap, no re-use	3	2	0	5	1	0.6	0.4	0	0.71	0.83
No swap, re-use	3	2	0	5	1	0.6	0.4	0	0.71	0.83
Neighbouring swap, no re-use	3	2	0	5	1	0.6	0.4	0	0.71	0.83
Neighbouring swap, re-use	3	2	0	5	1	0.6	0.4	0	0.71	0.83
Random swap, no re-use	3	2	1	4	0.8	0.6	0.4	0.2	0.67	0.73
Random swap, re-use	2	3	0	5	1	0.4	0.6	0	0.63	0.77

The confusion matrices show the correlation between the actual (B and M) and predicted (b and m) number of benign and malignant classifications. The statistics provide details of the true positive (TP), true negative (TN), false positive (FP) and false negative (FN) rates, in addition to the precision (P) and the F_1 measure (F_1)

shown in Table 10, which includes: the elements from the confusion matrices, the true and false positive and negative rates, the precision rate, and the result of the F_1 measure [38]. From these results, it is possible to see that the best chromosomes for the no swap and neighbourhood swap recombination strategies (with and without re-use) are capable of classifying 80% of the test images correctly, whereas the random swap strategy performs slightly worse, as the best chromosome is only capable of classifying 70% of the test images correctly. Looking at the classification breakdown for the chromosomes that reached 80% accuracy, it is possible to see that although they classified 20% of the test images incorrectly, these 20% were classified as false positive, so it is possible to say that the classifier always identified a malignant image.

However, the worst chromosome classifier in both the no swap and neighbourhood swap recombination strategies is far worse than the respective chromosome in the random swap recombination strategy. Overall, the majority of the chromosomes for all three recombination strategies are only capable of classifying 50% of the test images correctly. This is not surprising, as most of the chromosomes have only been trained on the benign sections of the images, so have no concept of malignancy. For each recombination strategy, there only existed one or two chromosomes that produced the best classification result, so it is highly probable that these were the chromosomes that would have been tested on a section of the image which was classified by the radiologist as malignant.

As the majority of the single chromosomes only classified 50% of the test images correctly, it would be interesting to see if using the second classification approach, which involved voting amongst all the chromosomes for each section, improved the classification accuracy. The results for this classification approach for two different voting thresholds are shown in Tables 11 and 12. From the results, it can be seen that the voting threshold has a pronounced impact on the results. For a voting threshold of 1 (more than 1 chromosome classifies the image section as malignant), the random swap recombination strategy can classify 60% and 70% of the test images correctly, depending on whether re-use is used. However, with a voting

Table 11 Test image classification using multi-chromosome voting with a voting threshold of 1 ($V_{TH} = 1$)

	Confusion matrices				Statistics					
Recombination	Bb	Bm	Mb	Mm	TP	TN	FP	FN	*P*	F_1
No swap, no re-use	0	5	0	5	1	0	1	0	0.5	0.67
No swap, re-use	0	5	0	5	1	0	1	0	0.5	0.67
Neighbouring swap, no re-use	0	5	0	5	1	0	1	0	0.5	0.67
Neighbouring swap, re-use	0	5	0	5	1	0	1	0	0.5	0.67
Random swap, no re-use	1	4	0	5	1	0.2	0.8	0	0.56	0.71
Random swap, re-use	2	3	0	5	1	0.4	0.6	0	0.63	0.77

The confusion matrices show the correlation between the actual (*B* and *M*) and predicted (*b* and *m*) number of benign and malignant classifications. The statistics provide details of the true positive (TP), true negative (TN), false positive (FP) and false negative (FN) rates, in addition to the precision (*P*) and the F_1 measure (F_1)

Table 12 Test image classification using multi-chromosome voting with a voting threshold of 2 ($V_{TH} = 2$)

	Confusion Matrices				Statistics					
Recombination	Bb	Bm	Mb	Mm	TP	TN	FP	FN	*P*	F_1
No swap, no re-use	0	5	0	5	1	0	1	0	0.5	0.67
No swap, re-use	0	5	0	5	1	0	1	0	0.5	0.67
Neighbouring swap, no re-use	0	5	0	5	1	0	1	0	0.5	0.67
Neighbouring swap, re-use	3	2	1	4	0.8	0.6	0.4	0.2	0.67	0.73
Random swap, no re-use	2	3	0	5	1	0.4	0.6	0	0.63	0.77
Random swap, re-use	0	5	0	5	1	0	1	0	0.5	0.67

The confusion matrices show the correlation between the actual (*B* and *M*) and predicted (*b* and *m*) number of benign and malignant classifications. The statistics provide details of the true positive (TP), true negative (TN), false positive (FP) and false negative (FN) rates, in addition to the precision (*P*) and the F_1 measure (F_1)

threshold of 2, the random swap with re-use strategy does not perform well any more but the neighbourhood swap with re-use and the random swap without re-use strategies both classify 70% of the test images correctly. Overall, it may be possible to deduce that the best performing recombination strategy when chromosome voting is used, is the random swap without re-use, as it appears to be more robust to changes of the voting threshold and never classifies any of the test images as false negative, so a malignant test image is always detected.

An example result is shown in Fig. 9. Each figure represents an image part (8×8 pixel area), the number 1 indicates malignant tissue in that area and the number 0 indicates benign tissue. The radiologists classification of malignancy is indicated by grey shading and a white oval in the figure representation and mammogram RoI, respectively.

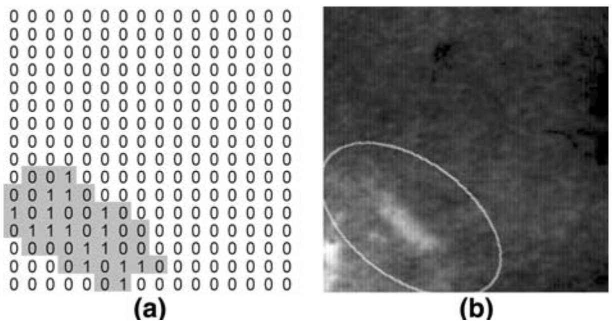


Fig. 9 An example of a Multi-chromosome CGP classification (**a**), for the corresponding region of interest in the mammogram (**b**). A "1" indicates the classification of malignant tissue by the Multi-chromosome CGP and "0" benign tissue, for the respective part in the region of interest. The *gray shading* and *white outline* depict the radiologist's classification of malignancy in **a** and **b**, respectively

7 Conclusions and future work

Parallel and distributed methods for evolutionary algorithms have concentrated on the advantages of using multiple populations of genotypes. Each individual genotype encoding a potential solution to the problem. We refer to this as Single-chromosome evolution (SCE). In this paper we have proposed the parallelisation on the individual genotype itself into a collection of chromosomes. We call this Multi-chromosome evolution (MCE). Such an approach allows one to re-examine the role of evolutionary operators and selection methods. In addition to mutating genes or recombining genes we can recombine, shuffle or select at a whole chromosome level.

Certain problems are naturally amenable to MCE. One such class of problems is digital circuit with multiple outputs as one can assign each output its own separate chromosome. We investigated both SCE and MCE on seven digital circuit problems using Cartesian Genetic Programming. We found that MCE could evolve solutions much faster than SCE. The speedup varied from 3 to 392 times faster depending on the benchmark problem. In the case of a small ALU with seventeen outputs, we were unable to evolve a correct solution with SCE, whereas we could evolve a fully correct solution using MCE.

When MCE was used for the digital circuit problems we used a generalisation of a (1+4) evolutionary strategy in which each chromosome of the parent genotype is formed of fittest chromosomes from the population. This acts like a enhanced form of recombination. The new population is formed by mutational offspring of the best possible combination of chromosomes from the previous population.

Having established that MCE has distinct advantages on multiple output digital circuit problems we carried out a case study of MCE on a different sort of problem. One in which there was not multiple outputs, but a problem that could naturally be divided into many parts, namely, the classification of mammograms. MCE can readily be applied to this problem by assigning an independent chromosome to each part of the image (in our case 256 parts). Unlike the digital circuit problem where chromosomes are dedicated to particular outputs, we can usefully swap chromosomes that operate on one part of an image for ones that operate on a different part. We investigated a number of ways to carry out this swapping (for instance, random swapping, and neighbouring swapping) to ascertain which ones are most effective. In addition, because in the image task, each chromosome can be applied to any part of the image, even parts it was not trained on, we have the opportunity of utilising the entire collection in decisions. For instance after evolution we can apply all chromosomes to every image part and make decisions based on a majority verdict.

Our results presented have demonstrated that the method obtains good classification performance on the problem of deciding whether microcalcifications are malignant or benign. Given the limitations of the training and the test sets available, and the lack of any pre-processing stage, our results are very encouraging. The main limitation of the image database we used is the low number of usable images. To overcome this problem, new databases of mammograms are being sought through a collaboration with a hospital.

A meaningful comparison with other Computer-aided Diagnosis (CAD) techniques is difficult to achieve due to the different methods used to assess performance. These issues are considered fully in [29] and will be investigated in future work.

Employing multi-chromosomes as population members, where chromosomes can be evaluated independently, offers the prospect of further research to explore the interaction and optimisation of multi-chromosomes in combination with other parallel and distributed evolutionary techniques.

Acknowledgments The authors would like to thank Eddy Munday for his work in preparing the regions of interest from the mammograms used in the experimental work described in this paper.

References

1. Center for Health Care Technologies Livermore. Lawrence Livermore National Library/UCSF Digital Mammogram Database. Livermore, CA, 1995
2. D. Andre, A. Teller, Evolving team darwin united, in *RoboCup-98: Robot Soccer World Cup II*, ed. by M. Asada, H. Kitano (Springer, 1999), pp. 346–351
3. R. Cavill, Multi-chromosomal genetic programming. PhD thesis, Department of Electronics, University of York, UK, 2006
4. R. Cavill, S.L. Smith, A.M. Tyrrell, Multi-chromosomal genetic programming, in *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO)*, vol. 2 (ACM Press, Washington DC, 2005), pp. 1649–1656
5. R. Cavill, S.L. Smith, A.M. Tyrrell, The performance of polyploid evolutionary algorithms is improved both by having many chromosomes and by having many copies of each chromosome on symbolic regression problems, in *Proceedings of the 2005 Congress on Evolutionary Computation Conference (CEC)*, vol. 1 (2005), pp. 935–941
6. R. Chow, Genotype to phenotype mappings with a multiple-chromosome genetic algorithm, in *Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO)*, vol. 3102 of *LNCS* (Springer, Seattle, 2004), pp. 1006–1017
7. S. Christensen, F. Oppacher, An analysis of koza's computational effort statistic for genetic programming, in *Proceedings of the 5th European Conference on Genetic Programming (EuroGP)*, vol. 2278 of Lecture notes in computer science (Springer, 2002), pp. 182–191

8. J. Clegg, J.A. Walker, J.F. Miller, A new crossover technique for cartesian genetic programming, in *Proceedings of the 2007 Genetic and Evolutionary Computation Conference (GECCO)* (ACM Press, 2007), pp. 1580–1587
9. P. Collet, E. Lutton, F. Raynal, M. Schoenauer, Polar IFS + parisian genetic programming = efficient IFS inverse problem solving. Genet. Program. Evolvable Mach. **1**(4), 339–361 (2000)
10. D. Corne, E. Collingwood, P. Ross, Investigating multiploidy's niche, in *Evolutionary computing: selected papers from the AISB workshop*, vol. 1143 of *LNCS* (Springer, Brighton, 1996), pp. 189–197
11. D.B. Fogel, Evolving artificial intelligence. PhD thesis, University of California, San Diego (1992)
12. L.J. Fogel, A.J. Owens, M.J. Walsh, *Artificial Intelligence Through Simulated Evolution* (Wiley, New York, 1966)
13. J.C. Fu, et al., Image segmentation feature selection and pattern classification for mammographic microcalcifications. Comput. Med. Imaging. Graph. **29**(6), 419–429 (2005)
14. T. Haynes, S. Sen, D. Schoenefeld, R. Wainwright, in *Working Notes for the AAAI Symposium on Genetic Programming*. Evolving a Team (Cambridge, AAAI, 1995)
15. D.W. Hillis, Co-evolving parasites improve simulated evolution in an optimization procedure. Physica D **42**, 228–234 (1990)
16. J. Jiang, B. Yao, A. Wason, A genetic algorithm design for microcalcification detection and classification in digital mammograms. Comput. Med. Imaging. Graph. **31**(1), 49–61 (2007)
17. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, 1992)
18. S. Lee, C. Lo, C. Wang, P. Chung, C. Chang, C. Yang, P. Hsu, A computer-aided design mammography screening system for detection and classification of microcalcifications. Int. J. Med. Inform. **60**(1), 29–57 (2000)
19. S. Luke, C. Hohn, J. Farris, G. Jackson, J. Hendler, in *RoboCup-97: Robot Soccer World Cup I*, ed. by H. Kitano. Co-evolving soccer softbot team coordination with genetic programming (Springer, New York, 1997), pp. 398–411
20. S. Luke, L. Spector, in *Proceedings of the 1st Annual Workshop on Genetic Programming (GP)*. Evolving Teamwork and Coordination with Genetic Programming (MIT Press, Stanford University, CA, 28–31, 1996), pp. 150–156
21. H.B. Mann, D.R. Whitney, On a test of whether one of 2 random variables is stochastically larger than the other. *Ann. Math. Stat.* **18**, 50–60 (1947)
22. H.A. Mayer, M. Spitzlinger, in *Proceedings of the 2003 Congress on Evolutionary Computation Conference (CEC)*. Multi-chromosomal Representations and Chromosome Shuffling in Evolutionary Algorithms (IEEE Press, Canberra, 2003), pp. 1145–1149
23. J.F. Miller, in *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO)*. An Empirical Study of the Efficiency of Learning Boolean Functions using a Cartesian Genetic Programming Approach (Morgan Kaufmann, Orlando, 1999), pp. 1135–1142
24. J.F. Miller, D. Job, V.K. Vassilev, Principles in the evolutionary design of digital circuits—part I. Genet. Program. Evolvable Mach. **1**(1), 8–35 (2000)
25. J.F. Miller, S.L. Smith, Redundancy and computational efficiency in cartesian genetic programming. IEEE Trans. Evol. Comput. **10**(2), 167–174 (2006)
26. J.F. Miller, P. Thomson, in *Proceedings of the 3rd European Conference on Genetic Programming (EuroGP 2000)*, vol. 1802 of Lecture Notes in Computer Science. Cartesian Genetic Programming (Springer, Edinburgh, 2000), pp. 121–132
27. J.F. Miller, P. Thomson, T.C. Fogarty, in *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications*. Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study (1997)
28. J. Niehaus, W. Banzhaf, in *Proceedings of the Fifth European Conference on Genetic Programming (EuroGP)*, vol. 2610 of Lecture Notes in Computer Science. More on Computational Effort Statistics for Genetic Programming (Springer, 2003), pp. 164–172
29. R. Nishikawa, Current status and future directions of computer-aided diagnosis in mammography. Comput. Med. Imaging Graph. **31**, 224–235 (2007)
30. M. Oltean, C. Grosan, in *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*. Evolving Digital Circuits Using Multi Expression Programming (IEEE Press, Seattle, 2004), pp. 87–90
31. M. Oltean, C. Grosan, M. Oltean, in *International Conference on Computational Science*. Encoding Multiple Solutions in a Linear Genetic Programming Chromosome (2004), pp. 1281–1288

32. M. Oltean, C. Grosan, M. Oltean, in *Proceedings of the 4th International Conference on Computational Science (ICCS)*, vol. 3038 of *LNCS*. Evolving Digital Circuits for the Knapsack Problem (Springer, Krakow 2004), pp. 1257–1264
33. C. Ryan, in *In AAAI Fall Symposium Series on Genetic Programming Working Notes*. GP Robots and GP Teams: Competition, Co-evolution and Co-operation in Genetic Programming (AAAI, 1995), pp. 86–93
34. H. Schwefel, Kybernetische evolution als strategie der experimentelen forschung in der stromungstechnik. Master's thesis, Technical University Berlin (1965)
35. T. Soule, in *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO)*. Voting Teams: A Cooperative Approach to Non-typical Problems Using Genetic Programming (Morgan Kaufmann, San Francisco, 1999), pp. 916–922
36. T. Soule, in *Proceedings of the 2000 Genetic and Evolutionary Computation Conference (GECCO)*. Heterogeneity and Specialization in Evolving Teams, pp. 778–785
37. J. Torresen, in *Proceedings of the 5th International Conference on Evolvable Systems (ICES)*, vol. 2606 of Lecture Notes in Computer Science. Evolving Multiplier Circuits by Training set and Training Vector Partitioning (Springer, Trondheim, 2003), pp. 228–237
38. C. Van Rijsbergen. *Information Retrieval*. Butterworths (1979)
39. V.K. Vassilev, J.F. Miller, in *Proceedings of the 3rd International Conference on Evolvable Systems (ICES)*, vol. 1801 of Lecture Notes in Computer Science. The Advantages of Landscape Neutrality in Digital Circuit Evolution (Springer, 2000), pp. 252–263
40. G. Veni, E. Regentova, L. Zhang, Detection of clustered microcalcifications with susan edge detector, adaptive contrast thresholding and spatial filters, vol. 5112 of *LNCS*, pp. 837–843 (2008)
41. J.A. Walker, J.F. Miller, in *Proceedings of the 7th European Conference on Genetic Programming (EuroGP)*, vol. 3003 of Lecture Notes in Computer Science. Evolution and acquisition of modules in cartesian genetic programming (Springer, 2004), pp. 187–197
42. J.A. Walker, J.F. Miller, in *Proceedings of the 2005 International Conference on Evolvable Systems (ICES)*, vol. 3637 of Lecture Notes in Computer Science. Improving the Evolvability of Digital Multipliers using Embedded Cartesian Genetic Programming and Product Reduction (Springer, 2005), pp. 131–142
43. J.A. Walker, J.F. Miller, in *Proceedings of the 2005 Genetic and Evoluationary Computation Conference (GECCO)*, vol. 2. Investigating the Performance of Module Acquisition in Cartesian Genetic Programming (ACM Press, 2005), pp 1649–1656
44. J.A. Walker, J.F. Miller. Automatic acquisition, evolution and re-use of modules in cartesian genetic programming. *to be published in IEEE Trans. Evol. Comput.* (2008)
45. M. Walker, H. Edwards, C. Messom, in *Proceedings of the 10th European Conference on Genetic Programming (EuroGP)*, vol. 4445 of Lecture Notes in Computer Science. Confidence intervals for computational effort comparisons (Springer, 2007), pp. 23–32
46. M. Walker, H. Edwards, C. Messom, in *Proceedings of the 2007 Genetic and Evolutionary Computation Conference (GECCO)*. The Reliability of Confidence Intervals for Computational Effort Comparisons (ACM, 2007), pp. 1716–1723
47. F. Wilcoxon, Individual comparisons by ranking methods. Biomet. Bull. **1**, 80–83 (1945)
48. T. Yu, J.F. Miller, in *Proceedings of the 4th European Conference on Genetic Programming (EuroGP)*, vol. 2038 of Lecture Notes in Computer Science. Neutrality and the Evolvability of Boolean Function Landscape (Springer, 2001), pp. 204–217
49. T. Yu, J.F. Miller, in *Proceedings of the 5th European Conference on Genetic Programming (EuroGP)*, vol. 2278 of Lecture Notes in Computer Science. Finding needles in haystacks is not hard with neutrality (Springer, 2002), pp. 13–25