

Dominant and Recessive Genes in Evolutionary Systems Applied to Spatial Reasoning

Thorsten Schnier and John Gero

Key Centre of Design Computing
Department of Architectural and Design Science
University of Sydney, NSW, 2006, Australia
fax: +61-2-9351-3031
email : {thorsten, john}@arch.usyd.edu.au

Abstract. Learning genetic representation has been shown to be a useful tool in evolutionary computation. It can reduce the time required to find solutions and it allows the search process to be biased towards more desirable solutions. Learning genetic representation involves the bottom-up creation of evolved genes from either original (basic) genes or from other evolved genes and the introduction of those into the population. The evolved genes effectively protect combinations of genes that have been found useful from being disturbed by the genetic operations (cross-over, mutation).

However, this protection can rapidly lead to situations where evolved genes interlock in such a way that few or no genetic operations are possible on some genotypes. To prevent the interlocking previous implementations only allow the creation of evolved genes from genes that are direct neighbours on the genotype and therefore form continuous blocks.

In this paper it is shown that the notion of dominant and recessive genes can be used to remove this limitation. Using more than one gene at a single location makes it possible to construct genetic operations that can separate interlocking evolved genes. This allows the use of non-continuous evolved genes with only minimal violations of the protection of evolved genes from those operations.

As an example, this paper shows how evolved genes with dominant and recessive genes can be used to learn features from a set of Mondrian paintings. The representation can then be used to create new designs that contain features of the examples. The Mondrian paintings can be coded as a tree, where every node represents a rectangle division, with values for direction, position, line-width and colour. The modified evolutionary operations allow the system to create non-continuous evolved genes, for example associate two divisions with thin lines, without specifying other values. Analysis of the behaviour of the system shows that about one in ten genes is a dominant/recessive gene pair. This shows that while dominant and recessive genes are important to allow the use of non-continuous evolved genes, they do not occur often enough to seriously violate the protection of evolved genes from genetic operations.

1 Learning Genetic Representations

Evolutionary systems, where random and probability-driven choices play an important part, are strongly influenced by the representation used. Instead of trying to create a

neutral representation and to minimize any influence on the outcome of the process, previous work has shown that it can make sense to intentionally introduce a bias into the representation. This can improve the generation of solutions in optimization problems, and can also be used to bias the results of an evolutionary system so that it produces more desirable results.

Starting with a simple representation, a ‘conventional’ evolutionary system is used to create individuals according to an application-specific fitness function. At the same time, a process observes how the system is using the initial representation. If sufficiently strong patterns are observed, new, evolved genes that represent these patterns are produced and added to the representation (Schnier & Gero 1995). The new evolved genes are used by the system, and can again be incorporated into other, larger evolved genes.

Using the evolved genes can lead to faster problem solving times. The evolved representation can also be re-used for new applications, where the new solution is expected to share features of the original application (see e.g. Gero & Kazakov 1996).

If the initial system is given a set of examples, and the fitness is set to be a measure of the resemblance to the examples, the evolved coding that is created in this process will represent features of the examples. In a second step, this representation can be used to create new individuals that share features of the examples (Schnier & Gero 1996)

The function of evolved genes is the same in both cases: protection of a set of genes from disruption by evolutionary operations like mutation and cross-over.

2 Interlocking Evolved Genes

Genetic operations are used to permute the genetic material from one or two individuals, creating a new individual. Cross-over and mutation are the most common operations (see e.g. Michalewicz 1992).

Cross-over operations work by cutting the genotypes into two parts, and then swapping the parts of two parents. Evolved genes, however, can easily interlock and lead to situations where no point can be found where cutting is possible without destroying one or more evolved genes, Fig. 1(a). If the operation was allowed to cut through an evolved gene, the protection of gene groups would be violated and the evolved genes would lose their function.

In previous work, evolved genes have therefore been created by combining only directly adjacent genes on the genotype into higher-level genes. In a bottom-up process, this leads to large, continuous groups of basic genes that are represented by an evolved gene. These genes cannot interlock, and conventional genetic operations thus work with little change, Fig. 1(b).

This ‘simple evolved gene’ model has been shown to be applicable to a range of applications. However, in some applications its restrictions may be undesirable. Figure 1(c) shows an example: while there is no simple evolved gene that can be established, it is still possible to identify useful evolved genes with non-local neighbours.

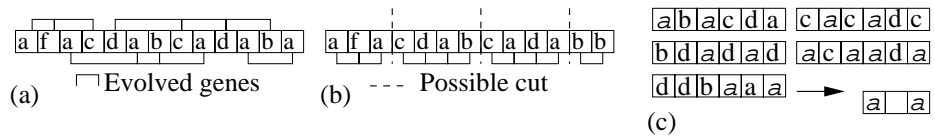


Fig. 1. (a) Interlocking evolved genes, (b) simple evolved genes, and (c) example for use of complex evolved genes

3 Using Dominant/Recessive Genes

To allow the use of arbitrary evolved genes, the genetic operations and the method used to create new individuals from evolved genes have to be modified. In doing this it is important to ensure that evolved genes will still be protected. The notion of diploid genetic representations with dominant and recessive genes has been used in genetic algorithms before, but with a very different purposes: a recessive set of genes can act as a memory of previously useful gene sequences, and help the genetic algorithm to adapt in applications with non-stationary fitness functions (Goldberg & Smith 1987, Ng & Wong 1995). Other work has shown that diploidity can help protect genetic algorithms against premature convergence (Greene 1996, Yukiko & Nobue 1994).

3.1 Cross-over

The cross-over operation consists of two steps: cutting the genotypes and re-assembling the two parts. To deal with interlocking evolved genes, the cutting operation has to be modified. It has to cut ‘around’ the evolved genes. Given a random cutting site, the complex genes spanning the cross-over site can be identified and the genotype cut so that the evolved genes remain completely on either of the two pieces. Figure 2 shows the cuts possible for the genotype in Fig. 1(a).

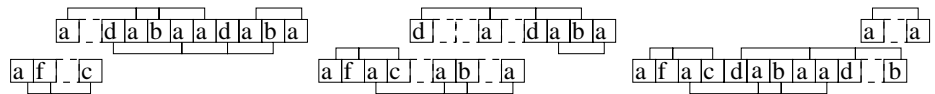


Fig. 2. Possible cross-over cuts for the genotype shown in Fig. 1(a).

The difficulty arises in the reassembly step: if at least one evolved gene spans the cutting site, the ends of the two sections that have to be combined into a new genotype will contain ‘holes’. If neither of the ends contains holes, the new genotype can be simply constructed by appending one segment to the other, Fig. 3(a). Similarly, if the holes in both ends happen to ‘fit’, both segments can be assembled in a ‘zipper’-like manner, Fig. 3(b).

In the general case, genes in one end will only partially overlap with holes in the other end. The solution suggested here borrows the notion of recessive and dominant

genes in sexual reproduction: allowing genotypes with pairs of genes in some locations of the genotype, with one gene shadowing the other, Fig. 3(c). When the genotype is transformed into a phenotype, one of the genes is expressed, the other ignored. Which gene is expressed is decided when the cross-over operation is executed. Possible criteria are: the gene of the larger evolved gene; the gene of the evolved gene with the higher fitness (see Section 4.3); or one of the two genes randomly selected.

Since some evolved genes are not entirely expressed in the phenotype, the protection of evolved genes is violated. However, this violation is small, since the rest of the evolved gene is still expressed. The important advantage of allowing overlapping, dominant and recessive genes, however, is that all evolved genes involved are still complete units and protected as evolved genes. In further genetic operations they will still be protected from being permuted by the genetic operation. It is also possible that the shadowed parts will be expressed again in an offspring after a cross-over or a mutation.

An interesting special case exists where the holes do not 'fit' but the values of the conflicting genes are equal. This 'continuity' between the two segments, Fig. 3(d), indicates that the new individual might have a higher chance of being successful.

An alternative option to allowing dominant and recessive genes is to append the segments so that 'holes' in the genotype remain, and fill the holes with random basic genes, Fig. 3(e). This adds a larger amount of random genetic material and would only be an option where a high mutation rate is otherwise used. However, it results in very long genotypes, especially once the evolved genes start getting relatively large, maybe spanning 20 or 30 genes. Equally unsatisfying is the option of just rejecting the cross-over: once a high percentage of evolved genes is reached, this would lead to a very strong restriction in possible cross-overs, reducing the performance of the system.

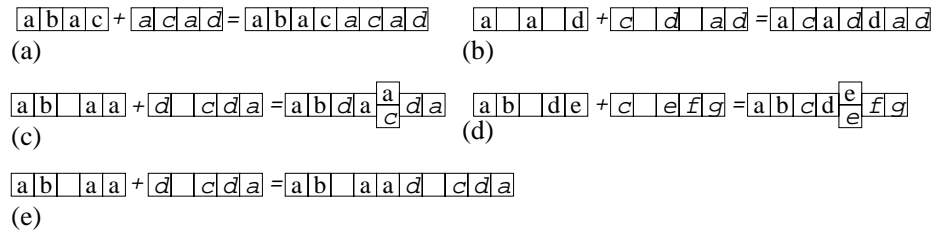


Fig. 3. Different ways to recombine two segments in a cross-over operation.

3.2 Mutation

In a mutation operation, genes in the genotype are randomly swapped with different genes from the current alphabet. While replacing a basic gene by a new basic gene is straightforward, replacing a complex evolved gene with another complex evolved gene is likely to create overlap. If an evolved gene is removed from a genotype, it leaves a pattern of holes; any different evolved gene will probably have genes at different places, and therefore will collide with genes in the genotype.

As with cross-over, using dominant and recessive genes provides a solution. At all places where the new evolved gene collides with an existing evolved gene, both conflicting genes are kept. The set of criteria from Section 3.1 can be used to decide which gene is the dominant and which the recessive. If a collision is caused by a basic gene in the genotype, the basic gene can simply be removed in favour of the evolved gene. To reduce the amount of overlap, one can optionally select the replacement gene as the best fitting one out of a random selection of evolved genes. Remaining holes are filled with basic genes, see Fig. 4. This figure also shows how a recessive gene can become expressed again as the result of a mutation.

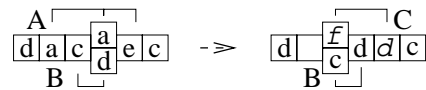


Fig. 4. Mutation using complex evolved genes: Evolved gene A is replaced by evolved gene C, one recessive gene of evolved gene B becomes visible, while another one becomes recessive.

3.3 Creating new individuals

If the evolved representation is re-used for a new application, a new initial set of individuals has to be created using this representation. If a set of complex evolved genes is randomly selected from the representation, it is unlikely that they can fit together without holes and overlaps. Re-ordering the evolved genes can minimize the overlaps, but for the remaining overlaps in the individuals the best solution is again to use dominant and recessive genes. Remaining holes can be filled with basic genes.

4 Application Example

The example in this section shows how complex evolved genes can be used to ‘learn’ features from an example set of Mondrian paintings and to produce new designs that share some of those features.

4.1 Coding Mondrian paintings

A large subset of Mondrian’s paintings can be described by recursive rectangle divisions, Fig. 5. These divisions can be mapped into a tree-shaped genotype. Any node in the tree contains a set of attributes that represent the position and direction of the division, the colour of one of the resulting areas, and the thickness of the separating line. As a result, the representation is a mixture of position independent (node) and position dependent (attribute) values.

Without evolved genes, the genetic operations are similar to those used in genetic programming (Koza 1992): the cross-over function swaps branches between two trees and the mutation operation replaces the value for an attribute with a random value. For

use with complex evolved gene, these functions are modified to allow dominant and recessive genes.

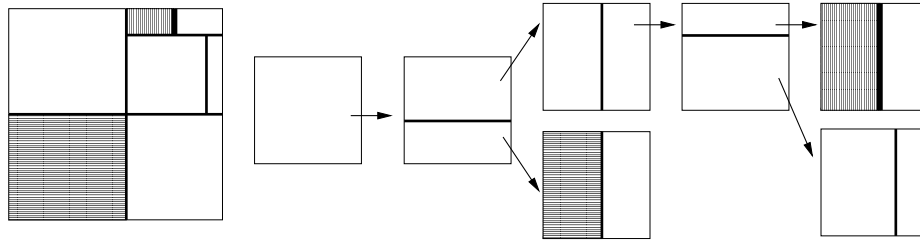


Fig. 5. Representation of a Mondrian painting by a series of rectangle divisions

4.2 Evolving Mondrian genes

In the first step a set of examples is given to the system, Fig. 6; the fitness is a measure of how close the individuals produced are to the examples. This is expressed in a set of Pareto-fitness values for each one of the examples, measuring the number of topologically correct and incorrect divisions, the exactness of the location of the divisions, correctness of colours and of line thicknesses.

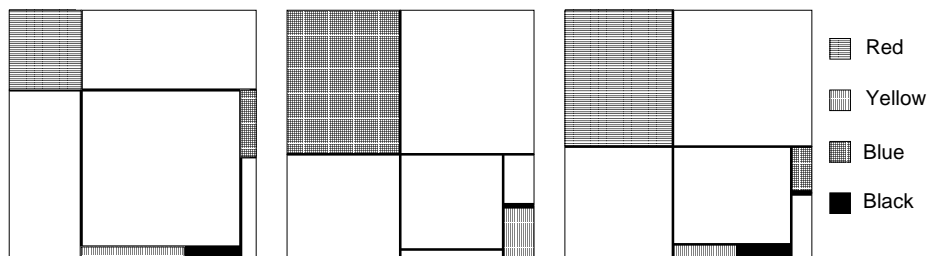


Fig. 6. Example paintings

4.3 Creating complex evolved genes

Ideally, one would want to allow any two genes on the genotype to be combined into an evolved gene. However, with large genotypes and a growing alphabet, this would lead to a combinatorial explosion of possible new evolved genes and would make it very hard to identify the best new gene. Fortunately, genes that are close on the genotype are more likely to be correlated in the fitness than genes that are far away. This allows a limited search by defining a 'search radius': two genes are only considered for gene

extraction if they are not more than n steps apart on the genotype. The size of this radius can be selected depending on the application. In this application, two basic genes are only considered for combination if they are on the same node or on directly adjacent nodes. Since every node consists of 4 values, this means that basic genes can be up to 8 values apart. In this way, evolved genes can grow to span a large number of nodes, with each node specifying the value of one or more of the attributes. Evolved genes can be combined with basic genes on any of the nodes that are part of the evolved gene and all nodes directly adjacent. Two evolved genes can be combined if the uppermost node of one of them is a node or directly adjacent to a node of the other evolved gene.

Even with a restriction to a certain search radius, the number of possible new genes can easily be too large. Fortunately, only a small fraction of the possible combinations will occur in the population at any given moment. Since any new evolved gene will obviously have to occur in the population, it is possible to restrict the search to the combinations existing in the population. A hash-table can therefore be used to allow efficient access to the different gene combinations. For the Mondrian application, the table usually contains about 3,000-4,000 entries, and the most common new genes occur about 700 - 800 times in a population of 1,000 individuals.

The fitness of a gene combination is calculated as the sum of the fitnesses of all individuals where this combination occurs. In the same way, a fitness can be calculated for all existing evolved genes. This fitness can be used to select evolved genes for mutation and generation of new individuals, and to decide which gene is the dominant where overlaps are produced.

Use of dominant/recessive gene pairs. Figure 7 is a plot of the average number of dominant/recessive gene pairs per individual during the creation of the evolved genes. It shows that the number of these pairs increases over the time, until about 10% of the genes are dominant/recessive pairs. As a comparison, about 5% of the genes in the final population are basic genes. Further analysis shows that about half the individuals in the final population (one quarter of all evolved genes in the population) have one or more gene that is part of a dominant/recessive gene pair. This shows that those pairs are an important factor in the evolutionary process. On the other hand, about 90% of all genes of evolved genes are expressed into phenotypes. The function of evolved genes to protect groups of genes from the genetic operations is therefore not substantially impaired.

Evolved genes. Figure 8 shows some evolved genes of different sizes created in this application. For example, the complex evolved gene in Fig. 8(a) specifies a division in the upper half of the rectangle (value 0 in first position), with a thin line (value 1 in third position). The other attributes are not specified by this evolved gene, and can be specified by basic genes or other evolved genes. Other evolved genes specify values at two or more nodes, Fig.s 8(c) and (d).

4.4 Creating new designs

The new representation can then be used to create individuals that are adapted to a new fitness function, but show features of the examples. In this example, the fitness function

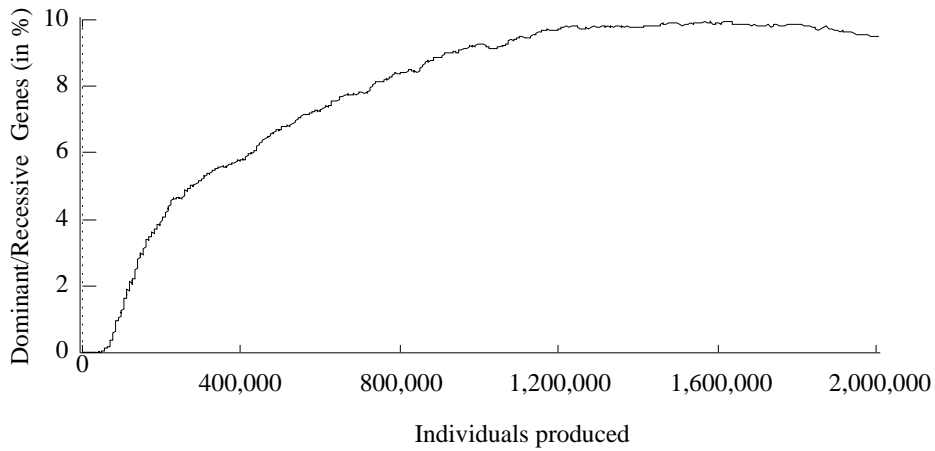


Fig. 7. Percentage of dominant/recessive gene pairs in the population

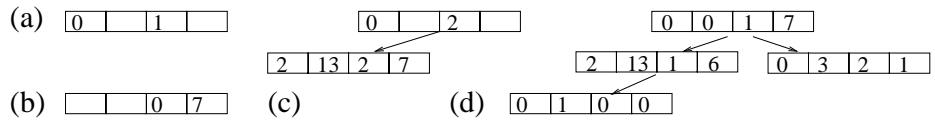


Fig. 8. Evolved genes created from examples, node values are (*direction – position – linewidth – colour*)

is very simple: seven rectangles, three colours and a white rectangle of a certain size in the top right corner. The intention is to make sure that any similarities to the Mondrian examples is caused by the use of the evolved coding, not by the fitness function.

The initial set of individuals is created from the evolved genes as described in Section 3.3. A relatively high rate of dominant/recessive gene pairs is to be expected as a result of an entirely random combination of evolved genes; in fact about 25% of the genes are dominant/recessive pairs. During the course of the evolution, this rate is reduced to about 15%.

Figure 9 shows the top two results of two different runs created with this fitness function without, Figs. 9(a) and 9(b), and with, Figs. 9(c) and 9(d), the use of the evolved coding. All results achieve perfect fitness, the runs without evolved genes converged slightly faster. While the evolved coding helps finding results with features that are present in the examples, it makes finding different features (the small white rectangle) slower. The goal of using the evolved genes, however, is to bias the results towards designs that appear similar to the examples. The success of this can be seen when the results in Figs. 9(c) and 9(d) are compared with the examples in Fig. 6; examples and results use similar colour combinations, line widths, and topological arrangements.

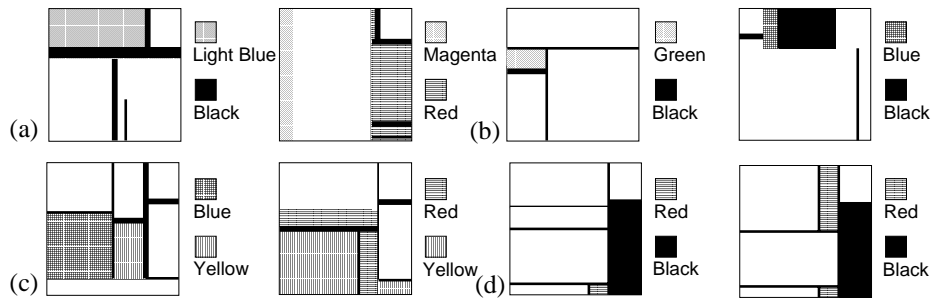


Fig. 9. (a) and (b) New individuals created without evolved genes, (c) and (d) individuals created with evolved genes. The colours used are listed next to the individuals.

5 Conclusion

This paper borrows the notion of dominant and recessive genes from sexual reproduction and applies it to evolutionary systems. It is used to resolve situations where one would like to preserve certain combinations of genes, but permutations from genetic operations lead to conflicts between two of those combinations.

Dominant/recessive genes allows duplicate genes to be kept in the genotype, where always one gene is expressed and the other gene is ignored in the construction of the phenotype, but preserved as part of the individual, and, importantly, as part of an evolved gene. None of the gene combinations therefore has to be destroyed. Moreover, the recessive gene is still part of the genetic material, and may be expressed again after further genetic operations.

Even though the paper focusses on the use of this idea in the context of evolved representation, it is applicable wherever the representation used leads to conflicts during genetic operations.

6 Acknowledgements

This work is supported by a grant from the Australian Research Council and by a University of Sydney Postgraduate Research Award.

References

- Gero, J. S. & Kazakov, V. (1996). Spatial layout planning using evolved design genes, in C. Dagli, M. Akay, C. Chen, B. Fernandez & J. Ghosh (eds), *Smart Engineering Systems: Neural Networks, Fuzzy Logic and Evolutionary Programming*, ASME Press, New York, pp. 379–384.
- Goldberg, D. E. & Smith, R. E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy, in J. J. Grefenstette (ed.), *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, pp. 59 – 68.

- Greene, F. (1996). A new approach to diploid/dominance in its effects in stationary genetic search, in P. J. A. Lawrence J. Fogel & T. Bäck (eds), *Evolutionary Programming V*, MIT Press, Cambridge, Massachusetts, pp. 171–176.
- Koza, J. R. (1992). *Genetic Programming. On the Programming of Computers by Means of Natural Selection*, Bradford Book, MIT Press, Cambridge, Massachusetts.
- Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, Berlin.
- Ng, K. P. & Wong, K. C. (1995). A new diploid scheme and dominance change mechanism for non-stationary function optimization, in L. J. Eshelman (ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Francisco, pp. 159–166.
- Schnier, T. & Gero, J. S. (1995). Learning representations for evolutionary computation, in X. Yao (ed.), *Australian Joint Conference on Artificial Intelligence AI '95*, World Scientific, Singapore, pp. 387–394.
- Schnier, T. & Gero, J. S. (1996). Learning genetic representations as alternative to hand-coded shape grammars, in J. S. Gero & F. Sudweeks (eds), *Artificial Intelligence in Design '96*, Kluwer, Dordrecht, pp. 39–57.
- Yukiko, Y. & Nobue, A. (1994). A diploid genetic algorithm for preserving population diversity - pseudo-meiosis ga, *Parallel Problem Solving from Nature - PPSN III*, Springer Verlag, Berlin, pp. 36–45.