Breaking the Symmetry of the Graph Colouring Problem with Genetic Algorithms

Anna Marino
Department Electronics and Computer Science
University of Southampton
Highfield, Southampton SO17 1BJ, UK
am97r@ecs.soton.ac.uk

Robert I. Damper
Department Electronics and Computer Science
University of Southampton
Highfield, Southampton SO17 1BJ, UK
rid@ecs.soton.ac.uk

Abstract

The graph colouring problem is a widely studied combinatorial optimisation problem with a highly symmetric solution space. Many heuristics and search algorithms are unattractive for general solution, i.e. they generate poor approximate solutions, because they were originally designed for specific problem instances. Population-based approaches like genetic algorithms do not provide a good alternative because of the danger of recombining good individuals from different regions of the search space (having different symmetries) to produce poor offspring. This paper presents a genetic algorithm that breaks the symmetry of the graph colouring problem by fixing the colours of the nodes in a large clique of the graph. Experiments have been conducted on both structured and random graphs to demonstrate the effectiveness of the approach.

1 Introduction

The ability to find quick and robust solutions to hard optimisation problems has been appreciated for long time since there are a vast number of practical applications. For example, a timetabling problem can be solved using a graph colouring one (de Werra 1997, 1999) as well as others. This paper focuses on the application of genetic algorithms to the graph colouring problem. Heuristics like tabu search (Hertz and de Werra 1987) or simulated annealing (Johnson, Aragon, McGeoch, and Schevon 1991) have been applied on this domain with some success. Other approaches include DSatur (Brélaz 1979) and a greedy algorithm (Turner 1988) which have now been outperformed by the use of hybrid techniques based on maximal independent sets (Culberson and Luo 1996; Galinier and Hao 1999). None of them takes implicitly into account one of the main features of the problem: namely, the presence of large symmetries in the solution space. That is, there exist a large number of different solutions with the same structure and the same cost – such that one solution can be mapped onto another simply by permutation of the colour labels. This is not a desirable feature when a search has to be applied. Ideally, we would like to have distinct representations for different solutions, although this approach is not always feasible when the space to be searched is not completely known.

This paper introduces a genetic algorithm that breaks the symmetry of the graph colouring problem by fixing the colours of the nodes in a large clique of the graph. Experiments have been conducted on both random and structured graphs. Section 2 illustrates the type of symmetries of the graph colouring problem while Section 3 gives some explanations of their effects on genetic algorithms. The symmetry breaking algorithm is introduced in Section 4 and results of its application are reported in Section 5. We find that performance is improved relative to a standard GA although the approach does not yet outperform the very best techniques so far reported. Section 6 discusses the limits of this approach and concludes.

2 Symmetries in Graph Colouring

Let $G(V, E)$ be a graph with vertex set V and edge set E. Let also $K = \{1, 2, \ldots, k\}$ be a set of colours. The vertex graph colouring problem (VGCP) consists of finding a partition of the graph G into k *colour classes* such that the number of edges with both endpoints in the same class is minimised. We will refer to such edges as *bad*. The smallest value of k for which G has no bad edges is called the chromatic number of the graph.

Assuming the graph G has n vertices (or nodes), the number $\Pi(n, k)$ of possible k partitions ($k < n$) with an arbitrary number of vertices in each class is given by:

$$\Pi(n,k) = \begin{cases} k^n & \text{(a)} \\ \sum_{i=0}^{k} (-1)^k \binom{k}{i} (k-i)^n & \text{(b)} \end{cases} \qquad (1)$$

as reported in Even (1973, pp. 56–69). In case (a), a class may be empty while, in case (b), all k classes need to be used. In both cases, the order of the colour classes is not relevant. The size of the solution space makes an exhaustive search impossible but, what is even less helpful, there is a high degree of symmetry for $\Pi(n,k)$. In fact, for any given partition, there exist $k!$ equivalent ones, which are obtained by a simple renaming of the labels of each class. Let us assume that solutions are represented by sequences of elements on the set K, where each element represents the colours of a node in the graph (we assume there is an ordering of the set V). So, for example, the sequence '114321' assigns colour 1 to the first, second and last node of a graph with six vertices, colour 4 to the third one and so on. The sequence '442134' represent an equivalent solution for the same graph since the first, second and last vertices are still grouped together, regardless of the name of the colour allocated to them. The degree of symmetry grows exponentially with the cardinality of the set K. Hence, it might be possible to improve the search by taking this into account.

The VGCP is known to be NP-complete and some features of its solutions space have been investigated by Grover (1992) while Cheeseman, Kenefsky, and Taylor (1991) found that it is possible to measure the hardness of problem in terms of the values of problem-dependent parameter. For the graph colouring problem, this parameter is represented by the edge probability p, which is defined as $p = \frac{2|E|}{n(n-1)}$. In other words, there exists a phase transition between instances of the problem that are easy to solve and those that are very complex.

3 Genetic Algorithms and Symmetric Spaces

Genetic algorithms (GAs) have been successfully applied to a variety of domains but the performance of the core technique (i.e. with swap/flip mutations and one point/uniform crossover) on the VGCP is quite poor unless they are hybridised with more specialised procedures (Marino, Prügel-Bennett, and Glass 1999; Galinier and Hao 1999). The major drawback has been identified as the recombination operator, since the relevant information about the partition of the graph for each parent is often lost at this stage. Figure 1 gives an example of the disruption generated by the recombination of two perfect solutions. The offspring turns out to be worse that its parents, mainly because the composition of the colour classes is not properly inherited, reflecting a *mismatch* of the colour labels. This phenomenon generates on average solutions with a higher associated cost while keeping a high diversity in the population. Despite the fact that the best-found solution can be retained across the generations using elitism, the convergence rate of the algorithm remains so low that good solutions may never be found.

It was observed that the label mismatch could be eliminated by selecting an appropriate permutation of one of the solutions (for more details see Marino, Prügel-Bennett, and Glass (1999)). In this way, the partition of the graph would not be changed but the cost of the offspring would not be worse than that of its parents. An issue to be addressed is the reduction/elimination of such symmetries, which can trap the search in local optima. For each possible partition, the

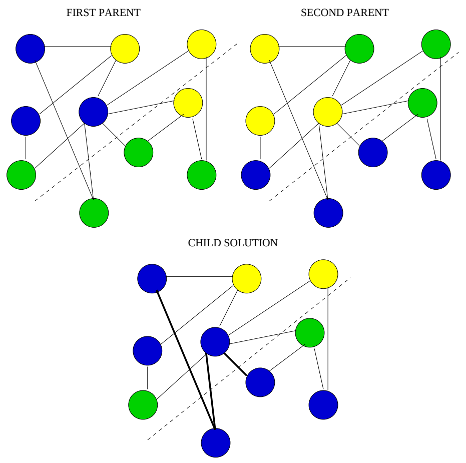

Figure 1: Example of disruption during the recombination of two perfect solutions. Colour clashes are shown by bold edges.

solution space contains $k!$ other equivalent ones which are simply differently represented. The actual number of distinct solutions in terms of structure is given by $\frac{\Pi(n,k)}{k!} = O(ke^n)$, which is still exponentially large even for small values of n and k.

4 Symmetry-breaking Approach

In this section, we describe a method that breaks the symmetry of the graph colouring problem by fixing the colours of some nodes. As pointed out in the previous section, a standard crossover operator is not appropriate in this domain because there is very little improvement of the solutions. Marino (2000a, 2000b) observed that cyclic permutations improve the performance of GAs on this domain. Experimental results showed that GAs applying cyclic permutations evolve populations where almost all individuals have the colour of the pivot node fixed (the pivot node is the one used to determine the offset of the permutation). This feature reduces the search space by one dimension. Would it be beneficial, then, if the colours of more nodes could be fixed somehow? If the size of the search space is reduced, the search is more likely to produce good results. Furthermore, if the colouring of a subgraph is fixed, the number of symmetries of the problem vanishes.

To fix the colour of some nodes in a graph, several issues need to be addressed: namely, how many nodes are to be allocated a fixed colour and which ones? Given that the colouration can use at most k colours, one may decide to freeze the allocation of k vertices of the graph. Vertices could be either randomly selected or chosen according to some predefined criteria (for example, the first k nodes in the graph representation). A random selection is obviously not a good choice since totally disconnected nodes may be allocated distinct colours without any further possibility of change. The benefits of fixing *a priori* the colouring of some vertices are maximal when all vertices are connected to each other, i.e. when they constitute a *clique* of the original graph. Unfortunately, it is not always possible to find a clique of size k in a given graph (in which case the chromatic number may be smaller).

We decided to fix the colours of the vertices in a clique of the graph of size at most k and evolve the colouration for the other nodes. An immediate consequence of this approach is that the symmetry described in Section 2 is broken because two solutions will represent the same partition of the graph if and only if they consist of the same sequence of elements. Therefore, the search of GAs is confined to a space of smaller dimension. A first step towards its application is to identify a clique of size at most k for the input graph by using a routine based on that of Turner (1988). All nodes in the clique are then assigned different colours and *marked* to avoid further modifications. Mutations and crossover apply to all other vertices of the graph but those marked.

For this application, GA operators have been modified to improve performance on the graph colouring problem. Standard mutations are replaced by more appropriate changes of the colour of a selected node. This approach has been already shown successful on this domain (Marino 2000a) since the cost of the solution cannot get worse when this operator is applied. The random selection of a typical mutation operator is replaced by a greedy selection of the colour which mostly reduces the cost of the whole solution. This procedure can be implemented with limited computational expense. In fact, it is possible to determine the new cost of a solution with a different allocation for a vertex by checking adjacent vertices of the mutated one only. Recombination is applied using a greedy uniform crossover. Once again, a local search mechanism was embedded in a genetic operator to enhance its performance. When two individuals are mated, the offspring inherits the colouring of the clique and all common genes with no changes. When two different values are encountered, the offspring receives the one from parent with the least associated local cost. This cost represents the number of bad edges in the subgraph induced by the adjacent vertices of the specified gene.

5 Experimental Results

The approach described in the previous section has been tested on a selection of random and structured graphs. These are available via `ftp` from file `instances.html` at URL `http://mat.gsia.cmu.edu/COLOR/`. The aim is to use GAs to generate good approximations for general graphs while keeping the algorithm as simple as possible. Instances represent benchmark problems with known chromatic number, except for the DSJC ones (from `instances.html`), for which simulations have been set using a number of colours that corresponds to their best known solution. This choice facilitates a qualitative and quantitative measure of the results found. Retention of the best solution is achieved by the application of elitism. Individuals are uniformly selected for mutations and recombination to avoid premature convergence of the population.

The aim of the simulations is to minimise a cost function which counts the number of bad edges. Thus, a cost value of zero indicates that the input graph can be coloured using k colours. Table 1 reports the number of vertices and edges of the selected graphs together with the number of colours they have been tested against.

Graphs	vertices	edges	k
`DSJC250.5.col`	250	31366	28
`DSJC1000.5.col`	1000	499652	83
`flat300_20_0.col`	300	21375	20
`flat300_28_0.col`	300	21695	28
`le450_15c.col`	450	16680	15
`le450_25c.col`	450	17343	25
`anna.col`	138	493	11
`homer.col`	561	1629	13
`queen8_12.col`	64	728	12
`queen9_9.col`	81	2112	10
`myciel5.col`	95	755	6
`myciel7.col`	191	2360	8

Table 1: Names of graph instances together with their number of vertices, edges and number of colours used for the simulations.

For each instance mentioned in the table, a set of five simulations with 50 initial random solutions and different seeds was run. Preliminary tests with populations of greedy solutions underperformed in many cases because of premature convergence of the population. This option was, therefore, discarded. Moreover, the algorithms turned out to be quite robust against randomness avoiding, thus, the need for a larger number of runs. The evolutionary process terminates either when a solution at cost zero is found or when the number of generations exceeds 10,000. This value represents a bound on the computational time allocated to the algorithm to find a solution. Given the particular nature of the operators used, mutations are applied with a rate of $p_m = 0.5$ while crossover takes place with probability $p_c = 0.1$. The choice of the population size and of p_m and p_c is not arbitrary but it represents the best combination of these parameters in a set of preliminary tests.

Figure 2 reports experimental results for three different graph instances, which are representative of the different performance behaviour observed. Several variants of GAs are mentioned in the rest of the paper. A GA with greedy mutations and the clique approach is referred to as CLIQUEGA as opposed to a standard GA with swap mutations and one point crossover, named SGA. A genetic algorithm using cyclic permutation is identified as CMGA. Results reveal that CLIQUEGA significantly outperforms a standard SGA in all cases. This is not surprising since we used more directed operators. These operators (i.e. the greedy mutation and uniform crossover) add local search mechanisms to increase the exploitation of the search space. All algorithms failed to produce the colouring of the graphs (with the specified number of colours) in many cases. Some instances seem to be easier to solve than others. In fact,

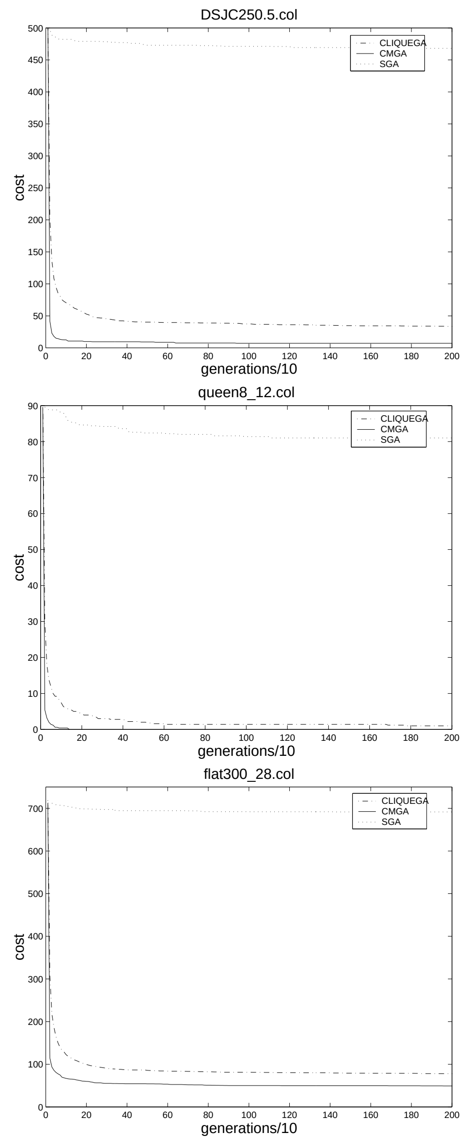

Figure 2: Average results for the best individual of the population on `DSJC250.5.col`, `queen_8_12.col` and `flat_300_28.col` using different GAs.

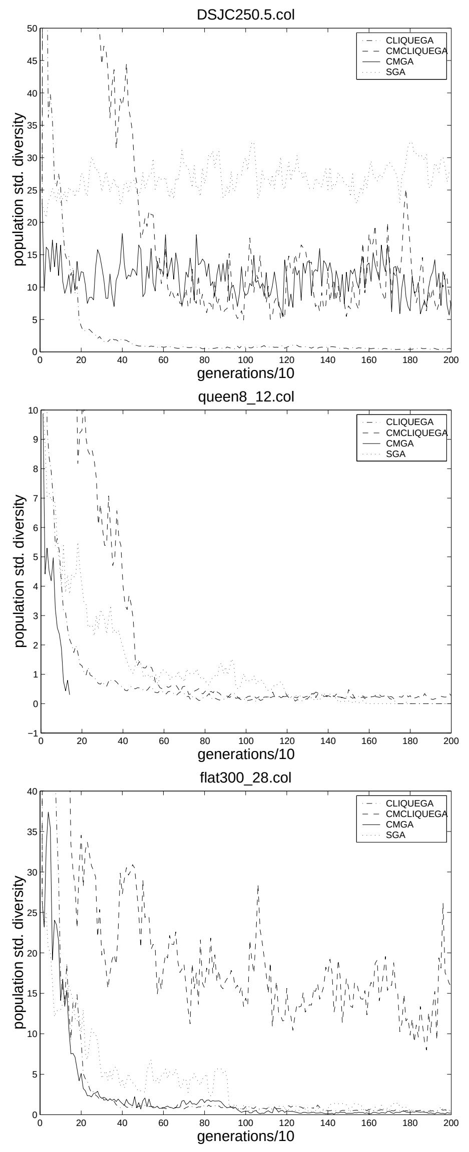

Figure 3: Average population standard deviation in CLIQUEGA, CMCLIQUEGA, CMGA and SGA on `DSJC250.5.col`, `queen_8_12.col` and `flat_300_28.col`.

a graph with a lower number of edges (i.e. with a smaller associated edge probability) is easier to colour since there is a larger number of possible partitions that satisfy the requirements of the problem. Furthermore, the phase transition of the graph colouring problem occurs when the value of the edge probability is between $7/n$ and $8/n$ (Eiben, van der Hauw, and van Hemert. J.I. 1998).

According to these figures, only the instances named `le450_25c.col` and `anna.col` have associated edge probabilities in that range while most of them have values above it and are, therefore, hard to solve. Surprisingly enough, Figure 2 reports a better performance of the CMGA in all cases. Why is that? The CMGA neither breaks the symmetry of the graph colouring problem nor guarantees any fitness improvement after recombination, but it still produces better solutions. Analysis of the simulation data reveals that the diversity of the population for the CLIQUEGA is negligible after 40 generations and this prevents further significant improvements of the solutions. Figure 3 compares the average population standard deviation for the three algorithms mentioned plus a fourth one, the CMCLIQUEGA, which will be later discussed.

The CMGA keeps variety in the population until convergence. Why does this not happen with the CLIQUEGA? A major cause of converge in the population is the combined action of the mutation and crossover. In both cases, the operators apply a local search where the selection is always done in a greedy way. Although this is also the case for the CMGA, the CLIQUEGA shows a worse performance because of the marked nodes. When a fairly good solution is found, the algorithm is very likely to reproduce it, even as a result of a crossover, when the other parent solution cannot contribute differently. In fact, since the CLIQUEGA breaks the symmetry of the graph colouring problem, there exist less possibilities for mixing different solutions while keeping the cost (if not lowering it). The absence of duplicated genotypes for the same solutions seems not to be beneficial on the basis of the experimental results. All GA variants with embedded local search mechanism (i.e. not the SGA) present a strong exponential reduction of the cost in the first 20 generations which is followed by virtually no further improvement. A complete explanation of this phenomenon is still not available but one hypothesis is that the algorithm gets trapped in a large orbit of a group acting on the set of colours and there is a low probability to escape from that orbit.

6 Discussion

The symmetry breaking algorithm introduced in the previous sections does not guarantee that there exists only one solution with a given cost. In fact, there might well exist different partitions of the graph that correspond to a similar cost. This can usually happen when the graph has lower degree nodes which can be allocated to different colour classes without involving any change in the cost.

A comparative measure of the performance of the algorithms discussed in Section 5 is not easy because of the formulation of the VGCP used for the investigation. However, the number of colours used for the simulations corresponds to the best-known solution for random graphs (the class is represented by the two DSJC instances) with edge probability $p = 0.5$ and the chromatic number of the graphs in the other cases. The symmetry breaking approach does not perform very well in any of the instances selected while showing a similar behaviour to the CMGA. In order to establish a better evaluation we should find what is the mininum value of k for which the CLIQUEGA finds a colouring of the graphs at cost zero. This investigation has not been done because we are more interested in speeding up the search process rather than finding the best possible solution. For a similar reason we did not perform simulations with a large number of generations (like for example the one reported by Galinier and Hao (1999)) since the efficiency of the algorithm would be significantly lower in this case.

To overcome the problem of low population diversity in the CLIQUEGA, we decided to embed the same mechanism adopted for the CMGA in the crossover, namely cyclic permutation. In a previous study, Marino (2000b) it was observed that they provide an easy and computationally inexpensive way of keeping a reasonable diversity in the population as well as having a good performance. The addition of cyclic permutations to the CLIQUEGA does not affect the vertices with fixed colouration though. We decided to apply the scheme only to the rest of the graph. Each time a crossover takes place, a cyclic permutation of the colours is applied to one of the parents with the aim of harmonising the similarity of the partition generated. This process will, therefore, attempt to solve the graph colouring problem for the set of unmarked vertices of the original one. The fact that some vertices have already been coloured is taken into account only when the algorithms evaluates the cost of the solutions found. In principle, this approach should be more disruptive than the original CLIQUEGA since we cannot guarantee that the colouring of the subgraph induced by the set of unmarked nodes matches the rest of the graph without generating additional bad edges.

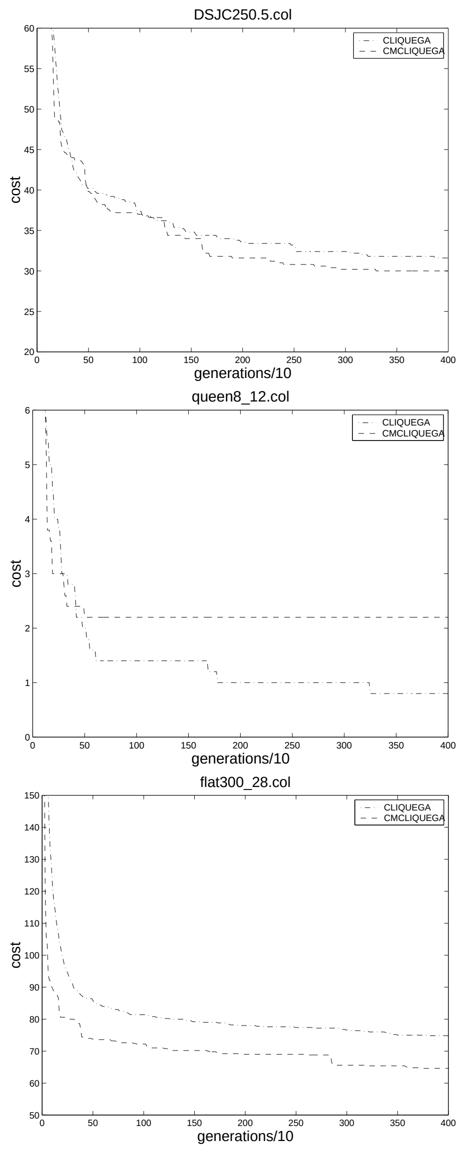

Figure 4: Comparison of the average best individuals for CLIQUEGA and CMCLIQUEGA on `DSJC250.5.col`, `queen_8_12.col` and `flat_300_28.col`.

Figure 4 compares the performance of the CLIQUEGA against the CMCLIQUEGA. The latter outperforms the former in almost all cases. The main reason for the slightly better results is the higher level of population diversity in the CMCLIQUEGA as can be observed in Figure 3. Therefore, the role of diversity is crucial in algorithms with high selective operators (i.e. mutation and recombination) that tend to reduce the differences between individuals. Unfortunately, diversity is not a sufficient condition to achieve a good performance, as the SGA shows, but it plays an important role.

The application of GAs to problems with large symmetries seems still to leave many open questions since the breaking of the symmetries did not by itself lead to remarkable improvements. We expected better results than those obtained. The CMGA still remains the best technique of those studied here. A complete understanding of its underpinning mechanisms is likely to contribute to the development of faster and more appropriate techniques for graph colouring problems. Symmetry breaking in this context offers a valuable ground for further investigations that aim both at refining this approach and at providing a better search method for spaces with a high degree of symmetry.

References

Brélaz, D. (1979). New methods to color vertices of a graph. *Communications of the ACM 22*, 251–256.

Cheeseman, P., B. Kenefsky, and W. Taylor (1991). Where the really hard problems are. In J. Mylopoulos and R. Reiter (Eds.), *Proceedings of 12th International Joint Conference on AI (IJCAI-91)*, Volume 1, pp. 331–337. Morgan Kauffman.

Culberson, J. and F. Luo (1996). Exploring the k-Colorable Landscape with Iterated Greedy. In *Cliques, Coloring and Satisfiability: Second DIIMACS Implementation Challenge*, pp. 245–284. Providence, RI: American Mathematical Society.

de Werra, D. (1997). The combinatorics of timetabling. *European Journal of Operational Research 96*(3), 504–513.

de Werra, D. (1999). On a multiconstrained model for chromatic scheduling. *Discrete Applied Mathematics 94*(1-3), 171–180.

Eiben, A., J. van der Hauw, and van Hemert. J.I. (1998). Graph coloring with adaptive evolutionary algorithms. *Journal of Heuristics 4*, 25–46.

Even, S. (1973). *Algorithmic Combinatorics*. Collier-Macmillan.

Galinier, P. and J.-K. Hao (1999). Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorics 3*(4), 379–397.

Grover, L. K. (1992). Local search and the local structure of NP-complete problems. *Operations Research Letters 12*(4), 235–243.

Hertz, A. and D. de Werra (1987). Using tabu search techniques for graph coloring. *Computing 39*(4), 345–351.

Johnson, D. S., C. R. Aragon, L. A. McGeoch, and C. Schevon (1991, May-June). Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. *Operational Research 39*(3), 378–406.

Marino, A. (2000a). Genetic search on highly symmetric solution spaces: Preliminary results. In *Theoretical Aspects of Evolutionary Computation*, pp. 391–411. Springer-Verlag.

Marino, A. (2000b). LAP-GA and CM-GA: Comparing genetic algorithms on the graph colouring problem. Submitted for publication.

Marino, A., A. Prügel-Bennett, and C. A. Glass (1999). Improving graph colouring with linear programming and genetic algorithms. In K. Miettinen, M. M. Mäkelä, and J. Toivanen (Eds.), *Proceedings of EUROGEN99*, Jyväskylä, Finland, pp. 113–118.

Turner, J. S. (1988). Almost all *k*-colorable graphs are easy to color. *Journal of Algorithms 9*(1), 63–82.