Genetic Algorithm for Graph Colouring: Exploration of Galinier and Hao's algorithm.

Celia A. Glass[1] and Adam Prügel-Bennett[2]

[1] *School of Mathematics, City University, London, EC1V 0HB, U.K.*

[2] *Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.*

November 9, 2001

Abstract

This paper examines the best current algorithm for solving the Chromatic Number Problem, due to Galinier and Hao (Journal of Combinatorial Optimization,1999, 3(4), pp 379-397). The algorithm combines a Genetic Algorithm with Tabu Search. We show that the algorithm remains powerful even if the Tabu Search component is eliminated, and explore the reasons for its success where other Genetic Algorithms have failed. In addition we propose a generalized algorithm for the Frequency Assignment Problem.

Key words: Graph Theory: Chromatic Number Problem, Frequency Assignment Problem; Heuristics: local search, genetic algorithms.

1 Introduction

The *Chromatic Number Problem* is one of the most studied Combinatorial Optimisation problems. The challenge is, given a graph, to find the least number of colours for which there is a colouring of the vertices of the graph in which no two adjacent vertices bear the same colour. This is most often implemented by using a conflict minimization algorithm. Given K colours, a colouring is sought which minimizes the number of conflicts (i.e. the number of adjacent vertices bearing the same colour). The best current algorithm for solving this problem is due to Galinier and Hao [1]. It is a hybrid colouring algorithm (HCA) which combines a new type of Genetic Algorithm with a Tabu Search. By contrast, most other recent improvements in tackling the Chromatic Number Problem have been achieved by incorporating maximum independent sets within existing non-GA approaches ([2–4]). Traditional Genetic Algorithms (GAs) gave poor results for graph colouring [5]. Hybrid algorithms which combine local search within a population based approach [4, 6–8] have enjoyed a certain degree of success although crossover has played only a small part in the success. In contrast, the success of Galinier and Hao is a direct consequence of their crossover operator.

Galinier and Hao tested out their algorithm extensively on benchmark graphs, thus demonstrating its competitiveness. For example, for a random graph in $\mathcal{G}_{1000,1/2}$ (graphs with 1000 vertices and adjoining edges each of which occurs with a probability 1/2), they achieved colourings with only 83 colours against the previous best result of 84.

In this short note we report on experiments to replicate Galinier and Hao's results, and explore the reasons behind the success of their approach. We also wish to highlight the use of their approach for more general problems such as Frequency Assignment in telecommunications.

2 Replication of HCA without Tabu Search

Galinier and Hao's hybrid algorithm, HCA, evolves a population using a novel crossover operator and a powerful local search method, Tabu Search, within a steady-state GA. At each generation, two members of the current population are selected and crossed to produce a child. Tabu search is applied on the child, which then replaces one member of the population. The crossover operator selects colour classes alternately from each of the two parents. The largest remaining class of the chosen parent is taken for the child, and vertices in that class are eliminated from both parents. When all K colour classes are established, the remaining vertices are assigned arbitrarily. Tabu Search plays a repair role, taking the new members of the population close to a local optimum with respect to vertex colouring. At the same time it decorrelates the population.

As well as developing a new crossover operator, Galinier and Hao also introduced some novel modifications within Tabu Search to give superior performance to previous Tabu Search algorithms. This rather obscures the question of whether their success was due to their Tabu Search, their crossover or a combination of the two. We implemented HCA without the Tabu Search component, relying instead upon a Vertex Descent algorithm to perform local search upon members of the GA population. The Vertex Descent algorithm is the standard one in which a neighbourhood arises from all the possible colourings of a single vertex. The full set of vertices is subjected to vertex descent repeatedly until no further improvement occurs for 100 iterations, although we cannot be sure that we have actually reached a local minima. We used a population size of 100 and a very mild selection procedure. A larger population, of 1000, was used for the largest random graphs. A new population is generated from the current one by retaining a single best member and adding to it the children, produced by HCA procedure, of 99 (or 999) arbitrarily selected pairs of the current population. Using a large population eliminates the difficulties of premature convergence of the GA, though a population size of 10 was adequate for smaller problems.

We repeated Galinier and Hao's experiments using this simplified algorithm. Our program, written in C++, was run on a 500MHz Pentium PC. Identical solutions to those reported in [1] were achieved for each of the seven, benchmark, problems, as illustrated in Table 1. The graphs are taken from the standard DIMACS challenge benchmark set [9]. The random graphs (DSJC*.*) have 250, 500 and 1000 nodes respectively, density 1/2 and unknown chromatic number. The remaining four graphs are each structured with known chromatic number. Two of the graphs are Leighton graphs (le*_*) with 450 vertices, and the remaining two are flat graphs with (flat*_*) 300 and 1000 vertices respectively.

Both our simplified version of HCA and the original algorithm did as well as the best known algorithm for that problem in all but one of the cases, namely le450_25c, and better than the best known results on the two largest graphs. For both the random and flat graphs with 1000 vertices the algorithms each reduced from 84 to 83 the number of colours required for a no conflict solution.

Graph	**Chromatic Number**	**Others**	**No. of Colours required GPB**	**HCA**
DSJC250.5	–	28	28	28
DSJC500.5	–	48	48	48
DSJC1000.5	–	84	**83**	**83**
le450_15	15	15	15	15
le450_25	25	25	*26*	*26*
flat300_28	31	31	31	31
flat1000_76	76	84	**83**	**83**

Table 1: Results on a number of test problems. Where known, the chromatic number is given for comparison. The minimum number of colours for which a colouring without colour conflict could be found, is presented for the different algorithms. HCA and GPB are the results of Galinier and Hao's Hybrid Colouring Algorithm and the adapted algorithm given in this paper. The best known results using other methods is also shown.

From our experiments it appears that the Tabu Search was not necessary for obtaining good performance of HCA. This is despite the enhanced search facilities of Tabu Search over Vertex Descent.

Without Tabu Search more iterations between crossovers and a larger number of crossovers are both to be expected. The extent of the difference is evident from the performance statistics presented for the two algorithms in Tables 2 and 3. Despite the apparent additional computational burder, the runtime of our algorithm on the benchmark problems, presented in Table 2, were not excessive. Curiously, although it took 50 hours to find a no-conflict colouring of DSJC1000.5 with 83 colours, all the other random graphs of the same size which we tested were quick to solve (?? about 3 hours2).

It should be said that although we used a standard vertex descent algorithm, it was carefully coded to run quickly, as described in Appendix A. The vertex descent algo-

Graph	K	**Parameters** P/L	**# successes**	**# generations**	**# iterations** ($\times 10^6$)	**Time**
DSJC250.5	28	100/100	3	118	11.7	9min
DSJC500.5	48	100/500	3	686	485	11hr
DSJC1000.5	83	1000/100	3	??	??	50hr
le450_15	15	100/100	3	11	1.9	55sec
le450_25	26	100/500	3	1571	2341	19hr
flat300_28	31	100/100	3	435	52.7	47.8hr
flat1000_76	83	100/200	3	305	177	13.3hr

Table 2: Performance statistics for GPB on different graphs. K represents the number of colours used. The parameters give the population size P and the number of descent cycles L between crossovers. Also shown is the number of successes out of 3, the number of generations (the number of crossovers is P times the number of generations), the number of iterations (i.e. the total number of decent moves), and the time taken for one complete run by the program on a 500MHz Pentium II processor.

Graph	K	**Parameters** P/L	**# successes**	**# crossovers**	**# iterations** ($\times 10^6$)
DSJC250.5	28	10/2000	9(1)	235	0.49
DSJC500.5	48	10/5600	5(5)	865	4.9
DSJC1000.5	83	10/16000	1	2015	28.4
le450_15	15	10/5600	6(4)	24	0.194
le450_25	26	10/1000	10	790	0.8
flat300_28	31	10/2000	6(4)	308	0.637
flat1000_76	83	10/16000	4(1)	1008	17.5

Table 3: Performance statistics for HCA on different graphs, taken from Galinier and Hao (1998), tables 4 and 5. As in the previous table we show the number of colours used and the parameters (population size and numer of local moves). Also shown is the number of successes (failures shown in brackets), the number of crossovers and the total number of local moves. Galinier and Hao gave no performance time so it is difficult to judge the cost of a tabu search move compared to a naive descent move.

rithm (run as a population, but without selection or crossover) was capable of finding colourings of $\mathcal{G}_{1000,1/2}$ using 94 colours within approximately 30 minutes. (It failed to find a 93 coulouring in 130 hours). Combined with kick-start it has found 89-colourings, as reported in [8]. However, using Galinier and Hao's crossover (GH-crossover) we always managed to find a 84-colouring and, given sufficient time, we usually find an 83 colouring. Clearly, the GH-crossover operator is significantly improving the search.

3 The role of large colour classes in the Genetic Algorithm

The idea behind crossover is to combine building blocks from different solutions to produce a child exhibiting the strengths of both parents. The difficulty with using a traditional GA crossover is that it does not preserve building blocks. This is, in part, caused by the permutation redundancy of the traditional graph colouring representation. The graph colouring problem is really a partitioning problem. That is, the graph needs to be partitioned into subgraphs with none (or few) edges in each subgraph. By treating the problem as a colouring problem we assign an arbitrary label to the partitions, and in so doing introduce a massive redundancy due to the labelling permutation symmetry. Traditional crossover does not respect this symmetry. That is, if we permute either of the parents and perform crossover we would get a very different child than if we had left the parent unpermuted. The achievement of Galinier and Hao is that they managed to find a crossover operator that overcomes the permutation problem. Permuting either of both parents before a GH-crossover does not change the child.

Removing the permutation symmetry is not, however, the only reason for the success of HCA. We experimented with replacing the GH-crossover with a new crossover where we first permuted one parent so that it was as highly correlated as possible with the other parent. (It is possible to do this efficiently by recognizing that the problem is a linear assignment problem—this is discussed in Appendix B.) However, this new crossover was no more effective than running the vertex descent algorithm with a kick-start to prevent the solutions getting trapped in local minima. Thus, HCA is doing more than just solving the permutation problem, they seem to have identified the important building blocks, namely the colour classes. These, after all, are the partitions of the graph. They are similar to independent sets, except that they can contain conflicts and so they are, in this respect, more general than independent sets (we return to this later).

It is still not obvious that GH-crossover should work. In the traditional independent set formalism many different combinations of independent sets are tried to find one combination which covers the whole graph. In GH-crossover only a single combination of colour classes are tried and then a local search is used to repair the damage. We would expect all the systematic damage or disruption caused by crossover to come from these unassigned vertices—the conflicts within the colour classes will be inherited from the parents and should be close to the average cost of the parents. However, the number of unassigned vertices is typically large—after choosing K colour classes there is still a substantial number of vertices that are unassigned. If we consider two random colourings, where the colour classes are chosen at random from either parent, we would expect 25% of the vertices to be left unassigned. In GH-crossover the colour classes are chosen in order of size. Empirically it was observed that around 18% of the vertices are unassigned at the beginning of the run. Later in the run the members of the population become correlated so that only about 5% of the nodes are unassigned by the end of a run. These unassigned nodes need to be assigned a colour and this will usually result in one or more conflict for each of these nodes. Thus, a child produced by GH-crossover will typically have many more conflicts that its parents. Remarkably, local search very effectively repairs this disruption—an empirical finding that would be hard to guess *a priori*.

This observation correlates with another empirical observation that comes from combining Vertex Descent with kick-starts. There, we found that if we start from a low conflict colouring and perform a perturbation by recolouring some randomly chosen fraction of the vertices, although this would result in a large increase in the number of conflicts, Vertex Descent would very rapidly find a colouring with the same small number of conflicts that we started with, almost every time. For example, when testing problems from $\mathcal{G}_{1000,1/2}$, we might spend several hours before finding an 89 colouring with only two conflicts, say. Perturbing this by recolouring 10% of the vertices would produce several hundred conflicts, yet Vertex Descent could usual find a two conflict colouring in a fraction of a second. This suggests that good solutions contain a fairly robust 'core'. GH-crossover enables a rapid and efficient search for good cores by combining different colour classes. Given a good core, local search can find a good solution, although it is very slow at changing the core structure. The GH-crossover and local search therefore complement each other nicely. In terms of a landscape picture, we can envision the cores as corresponding to local optima each with a fairly large basin of attraction. Kick-starts would allow an algorithm to explore nearby basins of attractions with a similar number of conflicts. GH-crossover allows large jumps to new local optima, but the good building blocks (partitions of the graph) are preserved. Although this is an attractive picture which is consistent with our observations, we admit that it is only hypothetical. We cannot be sure if we have reached a local optimum as the search space is too large to explore systematically. Furthermore, we have only a vague notion of what constitutes a building block—whether this can be usefully formalised remains an open question.

Using colour class partition actually has a significant advantage over independent sets, as the former does not have to be free from conflicts. This means that the GH-crossover can be used to find colourings below the chromatic number that have a small number of conflicts. In addition, it can be applied to weighted graph colouring problems. This opens up new application area such as the Frequency Assignment Problem.

4 Application to related problems

The Frequency Assignment Problem arises in the context of allocating radio frequencies to transmitter stations so as to minimize the total interference in the network. One variant of the problem is to depict frequencies as colours and transmitters as vertices with potential interference resulting from the same frequency being broadcast from two adjacent transmitters reduced to a penalty weight along the corresponding edge. The range of problems of this nature are described in [10]. In the basic version of the problem the number of colours is fixed, and the total sum of weights on those edges which connect vertices bearing the same colour is to be minimized. Many of the best methods for tackling the Chromatic Number Problem do not adapt well to this context, as discussed in [8]. In particular, the involvement of maximum independent sets in a solution method, which has proved so successful for the Chromatic Number Problem, is unlikely to produce great benefits. However, the HCA algorithm is ideal. The GH-crossover operator makes perfect sense in this more general context as it makes no reference to colour conflicts along edges, while vertex descent adapts naturally to take account of the weights of colour conflicts, as decribed in Appendix A.

Another variant of the Chromatic Number Problem for which the HCA approach is well suited, is when no colour class is allowed to be larger than a certain size. Other, more general, constraints on class size may also be accommodated.

5 Conclusion

From computational experiments, we conclude that the Genetic Algorithm (GA) component of the hybrid algorithm proposed and tested by Galinier and Hao [1] is strong enough on its own not to require Tabu Search. We show that when run, on the same test set, with a simple Vertex Descent mechanism in place of Tabu Search, their GA achieves equally good results.

The success of Galinier and Hao's Genetic Algorithm confirms that it crystallizes the essence of a graph colouring problem. By working with the partition of vertices into colour classes, it avoids $K!$-fold replication of solutions due to colour naming. Moreover, their algorithm provides a meaningful crossover operator, previously absent from the literature. Comparison with uniform crossover demonstrates that the way in which Galinier and Hao combine the classes of the partition affords a great advantage. Large colour classes appears to provide a better guide than independent sets in guiding local search.

An additional, highly attractive, feature of Galinier and Hao's approach is that it adapts easily to the Frequency Assignment Problem (in which the number of colours is fixed, edges may be weighed, and the sum of weighted colour conflicts is to be minimised) and to the Bounded Graph Colouring Problems (in which the number of vertices which may have any particular colour is restricted).

Appendix A: Implementation of a Vertex Descent algorithm

We employed the following fairly standard vertex descent algorithm. For a given colouring of a graph, each vertex is taken in turn and a colour selected for that vertex which gives the lowest cost. When there is more than one such colour a random choice is made. The algorithm cycles through all the vertices of the graph in order many times, until no further improvements occur.

Our implementation involves storing a matrix of the relative cost, $c(i,\mu)$, of colouring vertex i, with the particular colour, μ, while the colour of all other vertices remains unchanged. The relative cost terms are calculated as follows,

$$c(i,\mu) = \sum_{j=1|\mu=\kappa_j}^{k} w_{i,j},$$

where k is the total number of colours, and κ_j is the jth colour, and $w_{i,j}$ indicates the presence of edge (i,j), by taking the value 1 if there is such an edge and 0 otherwise. A *best* colour, μ, for a vertex i is one for which $c(i,\mu) \leq c(i,\nu)$ for all colours ν. From our matrix we compute a *colour list*, $L(i)$ of best colours for each vertex, i. To perform vertex descent for a vertex i, we choose one of the colours in the colour list $L(i)$. When a vertex changes colour we update the colour matrix for all its adjacent vertices and if necessary their colour lists. We also store the *colouring degree of freedom* of each vertex, defined as the length of this list minus one.

The implementation described above proved to be far more efficient than a more straight forward one. The overhead in using this method is that of computing the original cost matrix $c(i,\mu)$ and list $L(i)$, and then updating them after every colour change. However, checking to find a better colouring for a vertex becomes very quick. For any reasonable colouring, the vast majority of vertices have no colouring degree of freedom and therefore do not need to be updated. The overhead of the intial calculations is further minimized by the large number of times the set of vertices of the graph are scanned for improvement.

Observe that our implementation adapts well to weighted graphs. The term $w_{i,j}$ in the relative cost formula now refers to the weight on the edge (i,j).

Appendix B: Implementation of Uniform Crossover algorithm

In order to perform uniform crossover, in which no colour class is given preferential treatment, one must first pair the colour classes of the two parents, or equivalently to colour them. We need to permute the colours of one parent to make the two parents as closely coloured as possible. The measure of distance between parents is the usual Hamming Number, namely the number of changes in colouring of individual vertices required to get from one parent to the other. To minimize the distance between parents we produce the following linear assignment problem. A $K \times K$-cost matrix is constructed whose ij-th element is the number of vertices with colour i in the first parent and colour j in the second. An optimal colouring is achieved by applying the so called Hungarian Algorithm [11]. Uniform crossover is then performed, that is, vertices are coloured according to one or other parent arbitrarily. The colouring of the vertices of the child is more quickly performed with a vertex string than a partition representation.

References

[1] P. Galinier and J. K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

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

[3] A. Hertz and D. De Werra. Using tabu search techniques for graph coloring. *Computing*, 39:345–351, 1987.

[4] Craig Morgenstern. Distributed coloration neighborhood search. In David S. Johnson and Michael A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 335–357. American Mathematical Society, 1996.

[5] L. Davis. *Handbook of genetic algorithms.* Van Nostrand Reinhold, New York, 1991.

[6] Daniel Costa, Alain Hertz, and Olivier Dubuis. Embedding a sequential procedure within an evolutionary algorithm for coloring problems. *Journal of Heuristics*, 1:105–128, 1995.

[7] C. Fleurent and J. A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63:437–461, 1996.

[8] C. A. Glass and A. Prügel-Bennett. A polynomially searchable exponential neighbourhood for graph colouring. Available on the world wide web at `http://ecs.soton.ac.uk/`, 1998.

[9] David S. Johnson and Michael A. Trick, editors. volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, 1996. Available via ftp from `ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/`.

[10] A. Eisenblätter and A. Koster. FAP web - a website about frequency assignment problems. Available on the world wide web at `http://fap.zib.de/`, 2000.

[11] Carpaneto G. and Toth P. Algorithm 548: Solution of the assignment problem. *ACM Transactions of Mathematical Software*, 6(1):104–111, 1980.