Improving graph colouring with Linear Programming and Genetic Algorithms

Anna Marino †, *Adam Prügel-Bennett* †, *Celia A. Glass* ‡
† *Dept. of Electronics and Computer Science*
‡ *Dept. of Mathematical Studies*
University of Southampton
Southampton S017 1BJ, UK

Abstract

A major problem related to combinatorial optimization and population-based algorithms is the functional degeneracy of the search for problems with a high degree of symmetry. The population may in fact present multiple permutations of the same solutions which bound the exploration of the solution space. A genetic algorithm (GA) cannot generally cope with this problem unless additional mechanisms are included. The use of a simple GA together with the linear assignment problem (LAP) for the graph colouring problem has been investigated. The LAP algorithm is embedded in the crossover operator and generates the optimal permutation of colours within a cluster of nodes. This feature prevents offsprings generated by good solutions to be less fit than its parents. Simulation results show the successful application of this hybrid technique for the class of random graphs $\mathcal{G}_{1000,\frac{1}{2}}$.

1 Introduction

One of the most studied problems in combinatorial optimization is graph colouring. Many real problems can be modelled using graphs, and therefore having a fast and robust algorithm to generate optimal colouring is highly desirable. Unfortunately, due to the NP-completeness [1] of this problem, no general purpose technique is known and new approaches are a trade-off between accuracy, time complexity and plasticity.

Evolutionary techniques and in particular genetic algorithms have recently received attention since they can successfully solve a wide range of optimization problems with many local minima. The population-based approach of this technique allows large jumps in the search space. However, GAs have not proved successful for graph colouring because of the large degree of symmetry of the solution space. In fact, because of this symmetry mismatch, it is very unlikely to produce a fit offspring when combining two good solutions (which may be very similar up to a permutation). Thus, GAs are often considered an inappropriate approach for problems (such as graph colouring) with a highly degenerate objective function. In order to compensate for this degeneracy, advanced search techniques need to be applied.

This paper introduces a new crossover operator embedding linear programming techniques. It addresses the symmetry of the graph colouring by selecting the permutation of colours which mostly reduces the cost of the solution. Despite the fact the number of possible permutations is factorial with the number of colours, the optimization is performed in polynomial time.

1.1 The Graph Colouring Problem

Let $\mathcal{G}$ be a graph with connectivity matrix $W = \{w_{ij}\}$ where w_{ij} is the weight between nodes i and j. For weighted graphs $w_{ij} \in \mathcal{R}_+$ while for unweighted ones $w_{ij} \in \{0, 1\}$. A *k-vertex colouring* of a graph $\mathcal{G}$ is an assignment of k distinct colours to its vertices. The colouring is said to be *proper* when no two adjacent vertices have the same colour. A graph with a proper k-vertex colouring is said to be $k - colourable$. Ideally, we would like to find a proper colouring for $\mathcal{G}$ using the least number of colours. We say that a *clash* occurs when two connected nodes are allocated the same colour.

Since the complexity of this problem grows exponentially with the order of the graph, algorithms that provide approximate solutions are appropriate for large instances. Some of the techniques used are based on heuristics like simulated annealing [2] or tabu search [3] and probabilistic algorithms [4, 5]. More recently, some investigations have been conducted incorporating specific features of the problem in heuristics [6]. To illustrate the algorithm we consider the set of random graphs $\mathcal{G}_{1000,\frac{1}{2}}$ with n nodes and probability p of having an edge between any two nodes of the graph.

2 The Hybrid Algorithm

A general introduction to Genetic Algorithms may be found in [7, 8, 9]. A standard genetic algorithm with selection, mutation and recombination operators has been used. No particular attention has been paid to the selection criteria as the main goal is to investigate improvements coming from the other two operators. In fact, the mutation and the recombination operators have been developed to be highly biased towards improvements of the colouring of $\mathcal{G}$.

The aim of this paper is to test the use of such operators together with low selection pressure. The LAP algorithm is coded into the crossover operator of the genetic algorithm. Therefore, the evolutionary global search is added with a local search mechanism. Earlier studies [10, 11] suggested the successful outcome of such hybridization whose main feature is to produce quick and reliable improvements within regions of the search space.

2.1 The genome representation

There are several ways of coding a graph into a genome of a genetic algorithm. A major problem comes from the equivalence classes (with respect to the cost) of the solutions. Despite the fact that a unique mapping of the solutions is in principle possible, it does not help when the recombination operator is applied.

Early test simulations suggested the use of a naive representation, for which a sequential numbering of the nodes was necessary. Each gene codes the colour assigned to the correspondent node. Colours are assigned consecutive integer numbers in order to be able to select them. Therefore, saying that the i^{th} gene of the genome has a value of j means that the i^{th} node of the graph has been assigned colour j.

2.2 The fitness function

Many approaches to the graph colouring problem start with a proper colouring of the graph and try then to reduce the number of colours used. For this investigation the algorithm checks if the graph is k-colourable with a given k. The objective of the evolutionary process is then to minimize the number of clashes generated by a given allocation of colours. When no clashes are generated, the colouring is found.

2.3 The Mutation operator

This operator does not change or swap colours in the solution. A greedy choice of the colour for a given node has been applied instead. In particular, if i is a node of $\mathcal{G}$ and $N(i)$ a list of its adjacent nodes, then the colour σ_i is the one minimizing the number of clashes with respect to $N(i)$:

$$\sigma_i = \min_{\sigma \in K} \sum_{j \in N(i)} w_{ij} y_{ij}$$

where y_{ij} is a binary variable indicating if $\sigma_i = \sigma_j$.
When more than one choice is available, σ_i is randomly selected among all possibilities.

2.4 The Crossover operator

Let $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ be a partition of a graph and $K = \{1, \ldots, k\}$ a set of colours. Assume each node i is initially assigned a colour $\sigma_i \in K$ and we want to reorder colours in one of the subgraphs, say $\mathcal{G}_1$. Given the partition, the fitness of any solution can be split into three components according to [12]:

$$F_{tot} = F_{\mathcal{G}_1} + F_{\mathcal{G}_2} + F_{across}$$

where $F_{\mathcal{G}_i}$ is the fitness of each subgraph and F_{across} is the fitness across the two subgraphs. The crossover operator minimizes F_{across} while leaving $F_{\mathcal{G}_i}$ unchanged.

Let s_{ij} represents the sum of the clashes (across the partition) generated by changing colour i to j for all nodes in $\mathcal{G}_1$. The new allocation of colours for subgraph $\mathcal{G}_1$ has to minimize the sum of all s_{ij}. Despite the number of possible permutations is $|K|!$, this problem can be solved to optimality using the linear assignment problem formulation[1] [13]:

$$\begin{array}{lll}
\text{minimize} & z = \sum_{i \in K} \sum_{j \in K} s_{ij} p_{ij} & \\
\text{subject to:} & \sum_{i \in K} p_{ij} = 1 & \forall j \in K \\
& \sum_{j \in K} p_{ij} = 1 & \forall i \in K \\
& p_{ij} \in \{0, 1\} & \forall i, j \in K
\end{array}$$

where p_{ij} is a binary variable representing that colour i is changed to colour j. For this investigation $\mathcal{G}_2$ was chosen to be a **maximal zero-clashes** cluster of nodes[2]. This means it contains the maximum number of nodes of $\mathcal{G}$ whose allocated colours have no clashes. $\mathcal{G}_2$ was grown iteratively adding nodes of $\mathcal{G}$ to an initial random one such that $F_{\mathcal{G}_2} = 0$ at any time. The sexual implementation of the crossover mates subgraph $\mathcal{G}_2$ of one parent with the permuted version of $\mathcal{G}_1$ in the other one and vice-versa[3].

3 Experimental Settings and Results

Early tests indicated the superiority of a simple GA with elitism. The population is initialized with random colours for each node of the graph and has a size of 100; the

[1] Given a set of jobs M and a set of machines N, where each job i has a cost c_{ij} to run on machine j, the linear assignment problem consists of assigning each job to each machine at the least total running cost. In our case $M = K$ and $N = K$.

[2] This does not imply that $\mathcal{G}_2$ is an independent set for $\mathcal{G}$ since adjacent nodes may be considered.

[3] Due to different colourings $\mathcal{G}_2$ is not guaranteed to be a zero-clashes subgraph for the second parent.

probabilities for the mutation and recombination operators are respectively $p_m = 0.5$ and $p_r = 0.1$. Individuals are uniformly selected for breeding as it has been observed that ranked based selection leads to the premature convergence of the population. The algorithm stops either when a colouring is found or when the number of generations exceeds 5000. Simulations are averaged on 5 seeds and 10 instances. Graphs have been generated using a self-made software. Its random number generator[4] sets an edge between two nodes with probability p.

The investigation has been conducted on both weighted and unweighted instances of $\mathcal{G}_{1000,\frac{1}{2}}$ but, for clarity of exposition, plots will refer to unweighted graphs only. Results have been compared against the ones quoted in literature for the same problem and against a simple genetic algorithm with swap mutation and one point crossover. Two non population-based heuristics have been implemented as well. One iteratively applies the greedy selection embedded in the mutation operator while the other makes use of the same technique introduced for the crossover.

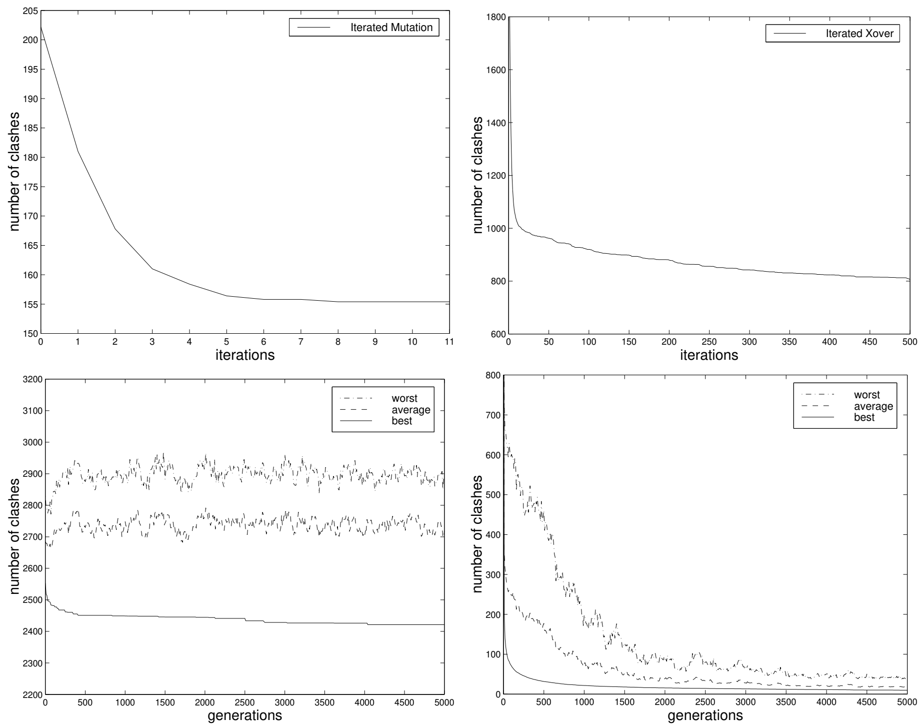

Figure 1: Average generated clashes for a 93-colouring of $\mathcal{G}_{1000,\frac{1}{2}}$ using an iterated mutation only (top left), iterated crossover only (top right), a simple GA with standard mutation and crossover (bottom left) and the hybrid GA+LAP (bottom right).

As depicted in the first two plots of figure 1, the iterated mutation heuristic provides quick improvements but results are far from being optimal. The average number of clashes does not go below 94.4 with 100 colours and 156.4 with 93 colours. On the other hand the iterated crossover heuristic is much slower and may converges to a proper colouring only asymptotically. In fact, the number of clashes is still quite high even after 500 iterations. The algorithm presented in this paper merges the two approaches using a population of solutions. A proper 100-colouring was found on average in less than 200 generations and a

[4]The algorithm is available from W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes in C*, Cambridge Press, 1989.

95-colouring with 1740. Figure 1 (bottom right) plots the average number of clashes for a 93-colouring which has not been always found. A greedy algorithm with the same number of colours generates on average 233 clashes. A comparison with figure 1 (bottom left) gives evidence of the substantial improvement of this hybrid technique against the standard GA whose performance is hardly noticeable. Despite the fact that the results shown do not outperform more specialized techniques which concatenate heuristic approaches like Tabu Search and those based on the Maximal Independent Sets formulation [6], this technique still provides comparatively good performance. The best colouring for $\mathcal{G}_{1000,\frac{1}{2}}$ instances reported in [6] uses 87 colours with Tabu Search (average is 93 though), 100-103 colours with Iterated Greedy and 113-119 with DSatur.

4 Conclusion and future work

The work presented suggests the idea that the application of evolutionary techniques to problems with a high degree of symmetry can actually be considerably improved incorporating more specific techniques such as the robust and quick operational research heuristics. Despite simulations on complex graph instances show that the search process becomes quite slow after the first 200 generations, a better parameter setting together with some modification of the core OR algorithm may actually help to speed up the search while preserving its robustness and generality. In particular, the crossover introduced in the paper should best perform on maximum cut graphs. Current investigations with other graph partitions are being made. Preliminary results on bi-partition turned out to be less powerful that the one in this paper. Random partitions and an approximate maximum cut one have been so far more promising ones since they address more directly the potentiality of the technique being used.

References

[1] M.R. Garey and Johnson D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freedman and Co., 1979.

[2] D. 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. *Operational Research*, 39(3):378–406, May-June 1991.

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

[4] J. A. Ellis and P. M. Lepolesa. A las vegas graph coloring algorithm. *The Computer Journal*, 32(5):474–476, 1989.

[5] Manvel. B. Extremely greedy coloring algorithms. In F. Harary and J. S. Maybee, editors, *Graphs and Applications*, pages 257–270, 1985.

[6] J. Culberson and F. Luo. *Cliques, Coloring and Satisfiability: Second DIIMACS Implementation Challenge*, chapter Exploring the k-Colorable Landscape with Iterated Greedy, pages 245–284. American Mathematical Society, 1996.

[7] L. Davis. *Handbook of Genetic Algorithms.* NY, Van Nostrand Reinhold, 1991.

[8] Z. Michalewics. *Genetic Algorithms + Data Structure= Evolutio programs.* Springer-Verlag, 1996.

[9] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[10] C. Cotta, J. F. Aldara, A. J. Nebro, and J. M. Troya. *Artificial Neural Networks and Genetic Algorithms*, chapter Hybridizing Genetic Algorithms with Branch and Bound techniques for the resolution of the TSP. Springer-Verlag, 1995.

[11] D. Srinivasan and A. Tattamanzi. A heuristics-based evolutionary approach to multi-objective generation scheduling. In *IEE Proceedings Part C-Generation, Transmission and Distribution*, November 1996.

[12] C. A. Glass and A. Prügel-Bennet. A new heuristic for the weighted graph colouring problem. Technical report, Faculty of mathematical studies, Southampton University, 1998. OR Preprint OR97.

[13] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons Inc., 1998.