Tree Search Methods versus Genetic Algorithms for Over-Constrained Graph Coloring Problems

Blaise Madeline[*] *Bertrand Neveu*[*]

[*] UNSA/INRIA/CERMICS
2004 route des Lucioles, BP 93, 06902 Sophia Antipolis, France
Email: {Blaise.Madeline, Bertrand.Neveu}@sophia.inria.fr

1 Context

Constraint satisfaction problems (CSP) are widely studied in a large area of computer sciences, and many problems can be treated as CSPs. Furthermore, many solving methods exist such as complete methods, local search (for example tabu search) and genetic algorithms (GA) [4]. The goal of this paper is to show that for some graph coloring problems, especially over-constrained, using genetic algorithms can be advantageous. In the first part, we show the results of using GA versus tree search methods, to find the minimal number of conflicts for coloring the n-queen problem (from Dimacs Challenge) with less colors than necessary. In the second part we introduce two graphs extracted from the wavelength division multiplexing (WDM) problem in all-optical networks [1], and we show the efficiency of GA to find the optimal (i.e. the minimal number of conflict in number of violated constraints) solution compared to tree search methods. We show that without special tuning of its parameters, GA outperforms these methods.

2 Using a Genetic Algorithm to solve over-constrained Graph Coloring Problems

Many methods can be combined with GA for solving graph coloring problems, like adaptive methods [5, 12], or hybrid methods [7], but in this paper, we only want to study the efficiency of standard GA with standard parameters. For these experiments, we used a GA Engine called AgCSP [3], which is dedicated to solve discrete CSPs. It works like a standard GA, but the chromosomes are integer coded instead of binary coded. It is easier to treat problems as graph coloring or random CSP using this coding when the domains of the variables are finite. The crossover is a 1-point crossover, the mutation is a per-allele mutation and a tournament selection is used. The evaluation function counts the number of violated constraints. For all the experiments the crossover rate is set to 1.

For the complete method, we used the ILOG solver library [9]. As the problem is a MaxCSP problem, thus an optimization problem, we used a *branch and bound* schema. To help the search, we used several heuristics. The variable selection heuristic is the *dom/deg* (minimum of the ratio between the domain size and the degree) from Bessière and Régin [2], the value selection heuristic is *min-conflict* (minimize the conflicts with the already instantiated variables, the ties being broken randomly) from Minton et al. [11], and the search method for the exploration of the search tree is the limited discrepancy search method (LDS) from Harvey and Ginsberg [8]. We used a filtering with *forward checking* and *inconsistency counts* introduced by Freuder and Wallace [6]. We also used another heuristic described later. The program using the tree search method for over-constrained graph coloring is called TSOGC in the rest of the paper.

2.1 The n-queen coloring problem

This is a well known problem from the Dimacs Challenge [10]. The goal is to color the squares of a chess board of size $n \times n$, such that two squares cannot be of the same color, if they are in the same line, the same column or the same diagonal. We especially studied the 8×8 board queen coloring problem (8QC).Our experiments on the 8QC show us that AgCSP has good results without special tuning. Furthermore, TSOGC must be guided by good heuristics to do as well or better.

The 8QC is a 9-coloring problem, but since we want to study over-constrained problem, we tried to color it with, 5, 6, 7, and 8 colors. For this coloring, minimal number of conflicts is respectively, 66, 40, 18, and for the 8 coloring problem we do not know the solution. We show our results in table 1, where MC is the minimum of conflicts in number of violations found by AgCSP and % is the success rate (i.e. percentage of runs for which a solution is found with MC conflicts or less). We used a mutation rate of 0.01, a population size of 40 with an elitism parameter set to 4. We stopped a run after 100 000 evaluations (60 seconds on a P-III 450). These results show us that AgCSP can give us good solutions in a good time (generally less than 30 seconds) but it is very difficult for it to reach the best solution. Using TSOGC gives better results than AgCSP when LDS method is used. Otherwise TSOGC cannot reach as good solutions as AgCSP after ten minutes, whatever its tuning. These results are not very impressive, but they encouraged us to find other larger and more difficult problems, to improve our approach.

Nb Color							
5		6		7		8	
MC	%	MC	%	MC	%	MC	%
66	11%	40	3%	<=21	3%	<=10	8%
<=67	47%	<=41	15%	<=22	7%	<=11	28%
<=68	77%	<=42	43%	<=23	24%	<=12	56%
<=69	91%	<=43	76%	<=24	58%	<=13	74%
<=70	100%	<=44	91%	<=25	81%	<=14	91%

Table 1: Percentage of runs of AgCSP for which a solution is found with MC conflicts or less for 8QC

2.2 Two WDM coloring problems

Finding hard graph coloring problems of reasonable size in order to do effective comparisons, is not simple thing. Especially when you want to know the chromatic number. Thus we have investigated for graphs containing odd cycles, in which the maximal clique is smaller than the chromatic number. We found them in problems associated to wavelength division multiplexing (WDM). In all-optical networks the vast available band-width is used through wavelength division multiplexing : a single physical optical link can carry several logical signals, provided that they are transmitted on different wavelengths (colors).

Given a set of requests and their routing on the WDM network, each request is associated to a path between two nodes of the network. The goal is to find a coloring of the paths such that two paths sharing an edge have different colors. Therefore, we built the conflict graph of the paths in which each node corresponds to a path, and two nodes are connected if their corresponding paths share the same edge of the network in the same way. Thus coloring the vertices of the conflict graph gives a coloring of the paths. In figure 1 (called Htree graph) each path represents one request, that we can generalize to n requests by path. Thus a node in the conflict graph represents a clique of n nodes in the graph coloring problem, and an edge represents a complete bipartite graph between the nodes. We show the resulting coloring graph with two requests by path in figure 2. The set of requests is called communication instance.

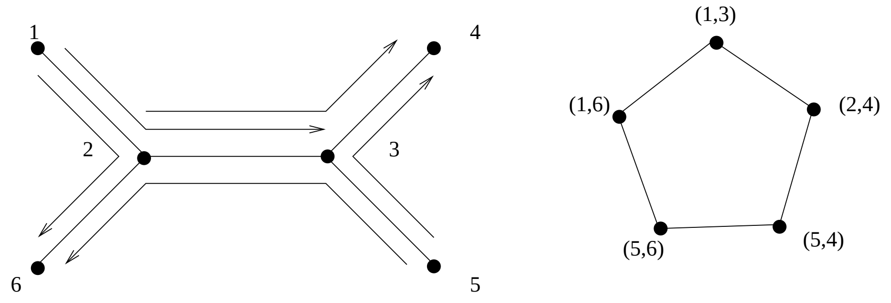

Figure 1: A routing for five requests in a Htree graph and its associated conflict graph

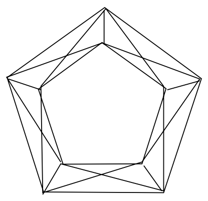

Figure 2: The associated graph coloring problem with 2 requests by path

We have defined a graph generator which, given a network, a communication instance and its routing, returns the coloring graph in Dimacs standard format [10]. For the Htree graph (figure 1), if the set of requests is low (up to 10 requests by paths), the problem is easy for TSOGC. But if the number of requests is high (above 20 requests by paths) the problem begins to be hard for complete methods. Thus we experiment with an instance of 30. The generated coloring graph has 150 nodes and 6675 edges, its maximal clique has 60 nodes, and it is colorable with 75 colors. We tried to color it with 60, 65, and 70 colors, with respectively 30, 20, 10 minimal number of conflicts in number of violated constraints. The results (see table 2) show us very good run time for AgCSP. The minimal number of conflicts is found in all cases in less than one minute. We run AgCSP with a 0.005 mutation rate, a population of 50, and elitism set to 4. TSOGC cannot find the minimal number of conflicts in less than 30 minutes, with only the *dom/deg*, *min-conflict* heuristics and LDS method. Thus, we added another heuristic which tries to break the symmetries by searching a clique and reduces the domain of the variables in the found clique. We called it *nosym*. With this new heuristic, the efficiency of TSOGC depends on the order of the nodes in the generated graph. If we choose the best tunings (as the first variable to proceed, the seed ...) and the generated graph with the best order for its nodes (i.e. the one that gives the best results) the best solution is found within less time than for AgCSP. But with the other generated graphs, the execution time is worse than AgCSP's results (table 2). No optimal solution is found after 10 minutes. Note that AgCSP found the optimal solutions whatever the order of the nodes.

We also made our tests on another graph that we called 3-Penta and which has more maximal cliques, and should be harder than Htree graph for complete methods. The figure 3 represents the conflict graph of 3-Penta with one request by path. We generated the coloring graph with 21 requests by path. It has 231 nodes and 8043 edges, the maximal clique is 42, and it is colorable with 53 colors. We tried to color it with 45 (minimal number of conflicts is 30), and 50 (minimal number of conflicts is 10).

Average time			
Nb color	60	65	70
AgCSP	19s	26s	43s

Table 2: Average number of seconds to find the minimal conflict for Htree graph with AgCSP

We show our results in table 3, the mutation rate is 0.005, the population size is 50 and the elitism was set to 8%. We stopped a run after 200 000 evaluations (800 seconds with a P-III 450). AgCSP found the minimal number of conflicts expressed as the number of violated constraints in 70% cases for 45 colors, and found good solutions (between best and best+2) for 50 colors in 70% cases, whatever the order of the nodes in the generated graph. TSOGC has the same behavior as for the Htree graph. Finding a good solution depends on the order of the generated graph. Furthermore, all the heuristics described before must be used, in particular LDS method and *nosym* heuristic to reach good solutions (close to minimal number of conflicts). But, in spite of these tunings (good parameters, all heuristics, and a "good" order of the nodes in the generated graph), TSOGC could not reach the minimal number of conflicts in 1 hour.

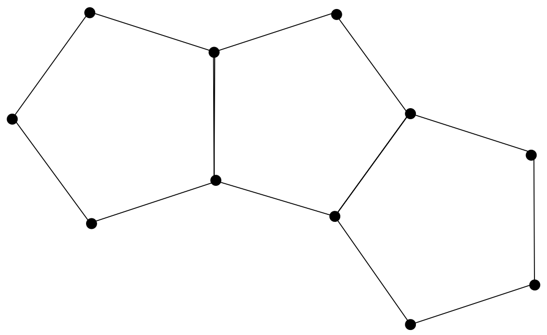

Figure 3: The 3-Penta conflicts graph

Nb color			
45		50	
MC	%	MC	%
30	70%	10	5%
<=31	100%	<=11	40%
		<=12	70%
		<=13	97.5%
		<=15	100%

Table 3: Percentage of runs of AgCSP for which a solution is found with MC conflicts or less for 3-Penta

3 Conclusion

This paper shows that genetic algorithms can be efficient, without special tuning or specialization, in finding good solutions in over-constrained graph coloring problems. It also shows that GA can outperform tree search methods, especially when the number of necessary colors is high. This result is interesting in particular when one has no time to look for a dedicated heuristic to guide a complete method. The graphs that we considered in our study are dense, and they have some maximal cliques which are highly connected between them. Furthermore the chromatic number is higher than the size of the maximal clique. Thus these problems are very difficult for tree search methods, even using good heuristics. Local search methods as tabu search or simulated annealing should also be tested in further experiments, and some testing should be done with hybrid methods and adaptive GAs.

Acknowledgements

We wish to gratefully acknowledge David Coudert and Hervé Rivano for their availability and enriching discussions about the WDM.

References

[1] B. Beauquier, J.-C. Bermond, L. Gargano, S. Perennes, P. Hell, and U. Vaccaro. Graph problems arising from wavelength-routing in all- optical networks. In *Proc. of the 2nd Workshop on Optics and Computer Science*, 1997.

[2] C. Bessière and J.C. Régin. MAC and combined heuristics: two reasons to forsake FC (and CBJ?) on hard problems. In *Proc. of CP'96*, volume 1113 of *LNCS*, pages 61–75, 1996.

[3] F. Didierjean. AgCSP, une bibliothèque de classes C++ pour le développement d'opérateurs génétiques pour CSP. Thèse en préparation.

[4] A. E. Eiben, P.-E. Raue, and Zsofia Ruttkay. GA-easy and GA-hard constraint satisfaction problems. In Manfred Meyer, editor, *Proc. of ECAI'94 Workshop on Constraint Processing*, pages 267–283, Amsterdam, 1994.

[5] A. E. Eiben and J. K. van der Hauw. Adaptive penalties for evolutionary graph coloring. In J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificial Evolution – Third European Conference*, pages 95–106, Berlin, 1997. Springer.

[6] E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.

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

[8] W. Harvey and M. Ginsberg. Limited discrepancy search. In Chris Mellish, editor, *IJCAI'95: Proc. of International Joint Conference on Artificial Intelligence*, pages 607–613, Montreal, 1995.

[9] ILOG. Ilog solver reference manual. Technical report, ILOG, 1998.

[10] D.S. Johnson and M.A Trick. Cliques, coloring and satifiability. In *Discr. Math. and Theoretical Comput. Sci., 2nd DIMACS Implementation Challenge*, volume 26 of *DIMACS Series*. 1993.

[11] S. Minton, M. Johnston, A. Philips, and P. Laird. Minimizing conflict: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[12] M.-C. Riff Rojas. A network-based adaptive evolutionary algorithm for constraint satisfaction problems. In S. Voss, S. Martello, I. Osman, and C. Roucairol, editors, *Meta-heuristics : Advances and Trends in Local Search Paradigms for Optimization*, pages 269–283. Kluwer Academic Publishers, 1998.