IMPROVEMENTS ON CONSTRUCTIVE GENETIC APPROACHES TO GRAPH COLORING

Geraldo Ribeiro Filho
LAC/INPE
Caixa Postal 515
12.201-970 São José dos Campos – SP
geraldo@lac.inpe.br

Luiz Antonio Nogueira Lorena
LAC/INPE
Caixa Postal 515
12.201-970 São José dos Campos – SP
lorena@lac.inpe.br

Key words: Graph Coloring, Genetic Algorithm, Combinatorial Optimization

Let $G = (V,E)$ be an undirected graph. A k-coloring of G is a partition of V into k subsets C_i, $i = 1,\ldots,k$, such that no adjacent vertices belong to the same subset. The *Graph-Coloring Problem* is to find k-coloring of G with k as small as possible. This optimal value of k corresponds to the chromatic number of G. It is well known that this problem is NP-hard [Garey & Johnson, 1978], and heuristics must be used for large graphs. Each vertex subset is an independent vertex set, and the coloring problem could be seen as a clustering problem to form independent vertex sets. Graph coloring is a very studied problem [de Werra, 1990; Korman, 1979] and efficient algorithms have been developed [Fleurent & Ferland, 1995]. Applications appear in scheduling (like timetabling) [de Werra, 1985; Leighton, 1979], frequency assignment [Gamst, 1986; Hale, 1980] and register allocation [Briggs et al., 1989]. The use of *metaheuristics* has produced the best results for a large class of graph instances. Johnson et al. [Johnson et al., 1991] and Chams et al. [Chams et al., 1987] applied *Simulated Annealing*. Costa and Hertz [Costa & Hertz, 1996] have created something based on *Ant Colony*. Friden et al. [Friden at al., 1989] and Hertz and de Werra [Hertz & de Werra, 1987] applied *Tabu Search* and Fleurent and Ferland [Fleurent & Ferland, 1994] applied *Hybrid Genetic Algorithm* with aggressive local search.

The *Genetic Algorithms (GA)* are very known, having several successful applications in *Combinatorial Optimization (CO)* problems [Fleurant & Ferland, 1995; Levine, 1993; Lorena & Lopes, 1996; Lorena &Lopes, 1997; Tam, 1992; Ulder et al., 1991]. *GA* is based on the controlled evolution of a structured population. The basis of a *GA* is the recombination operators and the schema formation and propagation over generations [De Jong, 1975; Golberg, 1989; Holland, 1975]. To obtain successful applications of *GA* to solve *CO* problems some characteristics of a classical *GA* have been adapted and redefined.

The *Constructive Genetic Algorithm (CGA)* approach (Ribeiro Filho, 1996) uses not only complete problem solutions, but also solution parts, known as schemata. The algorithm works with an initial population formed only by schemata. The schemata theory was for long the central point in classical *GA*, but has been less explored in recent years. A simple schema is not enough to represent a feasible solution for the coloring problem, as some vertices are not colored. New schemata or complete solutions are generated by schemata combination. Proportional fitting is used to evaluate schemata adaptation and *good* schemata are preserved. An evolution parameter eliminates schemata that do not satisfy a permanence criterion and the best schema found so far is kept. The process finishes with an empty population or when an iteration limit is reached.

A structure formed by elements, one for each graph vertex, was used as schema representation in the first CGA approach. Each element has the vertex color or a "*do not care*" symbol indicating the vertex is not colored in the present schema. The algorithm looks for a graph coloring that uses a number of colors *a priori* set. The population was kept ordered according the "*quality*" of the individuals, meaning more colored vertices and less conflicts (an edge linking same color vertices). The selection for recombination was made by taking an individual from the best ones in the population and another one from the whole population. Schemata recombination was made by merging the two selected schemata.

With representation modifications, the CGA was presented as a general heuristic for optimization problems (http://www.lac.inpe.br/~lorena/teseJC/CGA-tese.ps). Based on the representation used for the *p*-median problem (http://www.lac.inpe.br/~lorena/sbpo98/AGC-clust.ps), that can be generalized for clustering problems, an adaptation for graph-coloring problem was made so that it could be seen as a vertex-clustering problem. The basics of the first CGA application to the problem are the same. The selection process, the schema fitness evaluation, the evolution parameter are all the same. The main modifications were made in the schema representation, the population composition, a vertex-to-cluster assignment heuristic and a very aggressive mutation heuristic.

The new representation still uses structures with one element for each vertex, but only three symbols are possible. These are: the "*do not care*" symbol, indicating the vertices which are not assigned to any cluster; a symbol to indicate the vertex is a "*seed*" to form a cluster; and a third symbol indicating the vertices assigned to some cluster. The number of seed vertices is exactly the number of colors being used, or clusters being formed. The vertex-to-cluster assignment must be made by an appropriate heuristic.

Only schemata now compose the population. There is no complete solutions in the population. As complete solutions are generated, only the best one is kept. By the best one, we mean the one with less conflict edges. When the first schema is taken from the best ones in the population for recombination, a complete solution is created assigning all its vertices to a cluster, considering the clusters "*seeds*" already in it. This solution passes through a mutation process and is compared to the best complete solution found so far. As a second schema is taken and recombined with the one taken before, if another schema is generated, it is inserted into the population, but if a complete solution is generated, it passes through mutation and is compared to the best one. The recombination is the same used in the *p*-median application and merges the two schemata, but keep the number of seed vertices.

The vertex-to-cluster assignment uses an adaptation of a heuristic known as *Recursive Large First (RLF)* (Leighton, 1979) that has been compared to others and considered a very good one. This can be better understood using an example. Suppose we are looking for 3-coloring for a graph with ten vertices and the following adjacency matrix:

0111000000
1000001000
1001010000
1010101100
0001010110
0010100000
0101000001
0001100010
0000100100
0000001000

Let’s consider the following sets:

C_i is the set of vertices in the *i-th* cluster,
U_i is the set of all schema vertices adjacent to any vertex in C_i,
V_{sch} is the set of all the schema vertices, and
V_i is $V_{sch} - U_i$.

And let’s consider the following schema:

(#,1,0,1,#,0,0,#,1,0)	where	1 = Seed vertex
		0 = Vertex to be assigned
		# = Vertex not to be assigned

So, initially we have:

$C_1 = \{2\}$	$C_2 = \{4\}$	$C_3 = \{9\}$
$V_1 = \{3,6,10\}$	$V_2 = \{6,10\}$	$V_3 = \{3,6,7,10\}$
$U_1 = \{7\}$	$U_2 = \{3,7\}$	$U_3 = \{\ \}$

Now, take the vertex v in V_i, i=1,2,3 with the largest degree in U_i, i=1,2,3 and assign v to C_i. Then, update the sets C_i, V_i, and U_i. We obtain:

$C_1 = \{2,10\}$	$C_2 = \{4\}$	$C_3 = \{9\}$
$V_1 = \{3,6\}$	$V_2 = \{6\}$	$V_3 = \{3,6,7\}$
$U_1 = \{7\}$	$U_2 = \{3,7\}$	$U_3 = \{\ \}$

Repeating the process we have:

$C_1 = \{2,10\}$	$C_2 = \{4,6\}$	$C_3 = \{9\}$
$V_1 = \{3\}$	$V_2 = \{\ \}$	$V_3 = \{3,7\}$
$U_1 = \{7\}$	$U_2 = \{3,7\}$	$U_3 = \{\ \}$

The process continues until all sets V_i are empty.

At the end, in this example we will have the following clusters:

$C_1 = \{2,3,10\}$	$C_2 = \{4,6\}$	$C_3 = \{7,9\}$

The mutation process is based in the idea of taking the vertex with the largest degree as a "seed". It is applied to any complete solution generated and can be seen in the following pseudo-code:

Mutation Process

1: For each cluster
 Move the seed to the vertex with largest degree in the cluster
 Re-assign the vertices using the RLF approach
 Count conflicts and save the best in this loop
2: If the best found in the loop above is better than the original solution
 Replace the original by this best and return to pass 1
 Else
 Stop.

Computational tests were made with several instances taken from different groups: *book graphs* (Anna, David, Huck and Jean - each vertex represents a character and two vertices are connected if the corresponding characters encounter each other in the book); *game graphs* (Games120 – each vertex represents a team and two vertices are connected if they played each other during the season); *miles graphs* (Miles250, Miles500 and Miles750 – vertices representing cities are linked if the cities are close enough); *register graphs* (Musol_1, Musol_2, Zeroin_1 and Zeroin_2 - based on register allocation for variables in program code); *Mycielski graphs* (Myciel5, Myciel6 and Myciel7 - graphs based on the Mycielski transformation); *queen graphs* (Queen55, 66, 77, 88 and 99 - a graph with N^2 vertices, each corresponding to a square in NxN chess board, and two vertices connected if the corresponding squares are in the same row, column or diagonal).

The table bellow gives the computational results of the two approaches, the original CGA and the new one using clustering generalization and RLF based assignment. The table contains average numbers of vertices, edges and conflicts for each instance group. All the experiments were made with three runs for each instance, all of them for the optimal number of colors.

Group	Instances	Vertices	Edges	Original CGA	RLF CGA
Books	4	94.7	363.5	0	0
Games	1	120	638	0	0
Miles	3	128	1223.3	2.4	0
Register	4	204.8	3848.6	1.2	0
Mycielski	3	111	1117	1	0
Queen	5	51	753.2	24.3	0.5

The improvement in the quality of the results can be easily seen, especially for the *queen graphs*. Other tests must be made with larger graphs and also the CGA parameters must be analyzed.

References

Briggs, P.; Cooper, K.; Kennedy, K. and Torczon, L. (1989) Coloring heuristics for register allocation. In *ASCM Conference on Program Language Design and Implementation*, pp.275-284.

Chams, M.; Hertz A.and Werra, D. (1987) Some experiments with simulated annealing for coloring graphs. *European Journal of Operations Research* 32:260-266.

Costa, D. and .Hertz, A. (1996) Ants can color graphs. *Journal of the Operational Society* 47:1-11.

De Jong, K. A. (1975) Analysis of the behaviour of a class of genetic adaptive systems. *Ph.D. Dissertation - Department of Computer and Communication Sciences* - University of Michigan , Ann Arbor.

de Werra, D. (1985) An introduction to timetabling. *European Journal of Operational Research* 19(2):151-162.

de Werra, D (1990) Heuristics for Graph Coloring. In *Computational Graph Theory,* ed. Tinhofer, G. ; Mayr, E. ; Noltemeier, H. and Syslo, M., Springer-Verlag , Berlin, pp.191-208.

Fleurent, C. and Ferland, J. A. (1994) Genetic and Hybrid Algorithm for Graph Coloring, *Technical report* - Université de Montréal.

Fleurent, C. and Ferland, J. A. (1995) Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In David S. Johnson and Michael A. Trick, editors. Cliques, *Coloring, and Satisfiability: Second DIMACS Implementation Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*.

Friden, C.; Hertz, A. and de Werra, D. (1989) STABULUS: A technique for finding stable sets in large graphs with tabu search. *Computing* 42:35-44, 1989.

Gamst, A. (1986) Some lower bounds for a class of frequency assignment problems. *IEEE Transactions of Vehicular Technology* 35 (1):8-14.

Garey, M. R. and Johnson, D. S. (1978) *Computers and Intractability: a Guide to the Theory of NP-Completeness.* San Francisco, W. H. Freemann.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, New York.

Hale, W. K. (1980) Frequency assignment: Theory and applications. *Proceedings of the IEEE* 68(12):1497-1514.

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

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press. , Ann Arbor.

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

Korman, S. M. (1979) The graph-coloring problem. In Christofides, N. ; Mingozzi, A.; Toth, P. and Sandi, C. editors, *Combinatorial Optimization*: 211-235. JohnWiley & Sons, Inc., New York.

Leighton, F. T. (1979) A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards* 84: 489-506.

Levine, D. M. (1993) A genetic algorithm for the set partitioning problem. In *Proceedings of the 5th International Conference on Genetic Algorithms* , pp. 481-487.

Lorena, L.A.N. and Lopes, L.S. (1996) Computational Experiments with Genetic Algorithms Applied to Set Covering Problems. *Pesquisa Operacional* 16: 41-53.

Lorena, L.A.N. and Lopes, L.S. (1997) Genetic Algorithms Applied to Computationally Difficult Set Covering Problems. *Journal of the Operational Research Society* 48, 440-445.

Ribeiro Filho, G. (1996) Uma heurística Construtiva para Coloração de Grafos. *Master thesis*, INPE.

Tam, K. Y. (1992) Genetic algorithm, function optimization and facility layout design. *European Journal of Operational Research* 63:322-240.

Ulder, N. L. J. ; Aarts, E. H. L. ; Bandelt, H.-J. ; vanLaarhoven, P. J.M.and Pesch, E (1991) Genetic local search algorithms for the traveling salesman problem. In H.-P. Schwafel and R. Manner, editors, Springer-Verlag, *Proceedings 1st International Workshop on Parallel Problem Solving from Nature, Lecture Notes in Computer Science* 496: 109-116.