Constructive Genetic Algorithm and Column Generation: an Application to Graph Coloring

Geraldo Ribeiro Filho
geraldo@lac.inpe.br
UMC
Av Francisco Rodrigues Filho 399
08773-380 Mogi das Cruzes - SP - Brasil

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

Abstract

We present a combined use of Genetic Algorithms (*GAs*) and column generation to approximately solve graph-coloring problems. The proposed method is divided in two phases. The constructive phase builds the initial pool of columns using a Constructive Genetic Algorithm (*CGA*). Each column forms an independent set. The second phase solves by column generation the set covering formulation. The columns are generated solving weighted independent set problems. Some computational experience is given.

Key words: Column generation, Constructive Genetic Algorithm, Graph Coloring.

1. Introduction

Let $G = (V,E)$ be an undirected graph. A k-coloring of G is a partition of V into k subsets C_i, $i = 1,\dots,k$, 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 and 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 and Ferland (1994)]. 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. (1991) and Chams et al. (1987) applied *Simulated Annealing*. Costa and Hertz (1996) have applied *Ant Colony.* Friden et al. (1989) and Hertz and de Werra (1987) applied *Tabu Search* and Fleurent and Ferland (1994) applied *Hybrid Genetic Algorithm* with aggressive local search.

The *Genetic Algorithms (GA)* are very well known, having successful applications in *Combinatorial Optimization (CO)* problems [Fleurant and Ferland (1994), Levine (1993), Lorena and Lopes (1996), Lorena and 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), Goldberg (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 was recently proposed by Lorena and Furtado (1998) and Ribeiro Filho (1997). A typical *CGA* 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 a long time 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. A double fitting process 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 column generation approach to graph coloring was early studied by Mehrotra and Trick (1995) for implicit optimization of the linear program at each node of a branch and bound tree. The authors show that the method solves small to moderate size problems quickly.

2. CGA review

The *CGA* is proposed to address the problem of evaluating schemata and structures in a common basis. While in the other evolutionary algorithms, evaluations of individuals is based on a single function (the fitness function), in *CGA* this process relies on two functions, mapping the space of structures and schemata onto $\Re_+$.

2.1 Representation

The representation for structures and schemata uses three symbols. These symbols 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.

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 a 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 the following schema: (#,1,0,1,#,0,0,#,1,0). Where: 1 = seed vertex, 0 = vertex to be assigned and # = 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: $C_1 = \{2,3,10\}$ $C_2 = \{4,6\}$ $C_3 = \{7,9\}$

2.2. CGA modeling

Let C be the set of all structures and schemata that can be generated by the *0-1-#* string representation of *section 2.1.*, and consider two functions f and g, defined as $f : C$ ® $\Re_+$ and $g : C$ ® $\Re_+$ such that $f(s_i)$ £ $g(s_i)$, for all $s_i \in C$. We define the double fitness evaluation of a structure or schema s_i, due to functions f and g, as *fg-fitness*.

The *CGA* optimization problem implements the *fg-fitness* with the following two objectives:

(*interval minimization*) Search for $s_i \in C$ of minimal $\{g(s_i) - f(s_i)\}$, and

(*g maximization*) Search for $s_i \in C$ of maximal $g(s_i)$.

Considering the schema representation, the *fg-fitness* evaluation increases as the number of labels # decreases, and therefore structures have higher *fg-fitness* evaluation than schemata. To attain these purposes, a problem to be solved using *CGA* is modeled as the following *Bicriterion Optimization Problems (BOP)*:

$$\begin{array}{ll} Min & \{g(s_i) - f(s_i)\} \\ Max & g(s_i) \\ \text{subj. to} & g(s_i) \geq f(s_i) \\ & s_i \in C \end{array}$$

Functions f and g must be properly identified to represent optimization objectives of the problems at issue.

2.3. The fg-fitness

For the coloring problem the functions used are respectively

$$g(s_i) = \sum_{p=1}^{k} \left[\left(\left|C_{ip}\right| - 1\right).\left|C_{ip}\right|\right]/2\,,\quad f(s_i) = g(s_i) - \sum_{p=1}^{k} \left|E\left(C_{ip}\right)\right|, \text{ and } g_{\max} = mult.k.\left[\frac{\left(\lfloor n/k \rfloor - 1\right).\lfloor n/k \rfloor}{2}\right].$$

Where k is a pre-fixed number of colors, C_{ip} is the set of vertices receiving the color p on schema s_i (the notation $\left|C_{ip}\right|$ is the number of vertices in set C_{ip}), and $E\left(C_{ip}\right)$ is the set of edges with both terminal vertices in C_{ip} . The expression $(|C_{ip}|-1).|C_{ip}|/2$ gives the number of edges of a complete graph with $|C_{ip}|$ vertices.

Function *g(s$_i$)* can be interpreted as the total number of edges if k complete graphs of sizes $|C_{ip}|$ are considered. Function *f (s$_i$)* decreases this number by the number of edges actually linking vertices on the sets C_{ip}. When *f (s$_i$) = g(s$_i$)* the k sets C_{ip} are independent sets. To obtain the upper bound $g_{\max}$, first is considered divide the number of vertices n in k sets with approximately the same number of elements (the expression $\lfloor n / k \rfloor$ gives the large integer smaller than *n/k*), then the same procedure used for *g(s$_i$)* is applied, where the positive integer *mult* is considered to certify that $g_{\max} > \underset{s_i \in P_a}{Max}\, g(s_i)$.

2.4. The evolution process

The evolution process in *CGA* is conducted to accomplish the objectives (*interval minimization and g maximization*) of the *BOP*. At the beginning of the process, the following two *expected values* are given to these objectives. A non-negative real number $g_{\max} > Max_{s_i \in X}\, g(s_i)$, that is an upper bound to *g(s$_i$),* for each $s_i \in$ X, and the interval length $d\, g_{\max}$, obtained from $g_{\max}$ using a real number 0 < *d* ≤ 1.

The evolution process is then conducted considering an adaptive rejection threshold, which contemplates both objectives in *BOP*. Given a parameter a ³ 0 , the expression

$$g(s_i) - f(s_i) \ ³ \ d g_{\max} - a \,.d[g_{\max} - g(s_k)] \qquad (2.4.1)$$

presents a condition for rejection from the current population of a schema or structure s_i.

The right hand side of *(2.4.1)* is the threshold, composed of the expected value to the interval minimization $d\, g_{\max}$, and the measure $[g_{\max} - g(s_k)]$, that shows the difference of *g(s$_i$)* and $g_{\max}$ evaluations. For $a = 0$, *(2.4.1)* is equivalent to comparing the interval length obtained by s_i and the expected length $d\, g_{\max}$. Schemata or structures are discarded if expression *(2.4.1)* is satisfied. When $a > 0$, schemata have higher possibility of being discarded than structures, as structures present, in general, smaller differences $[g_{\max} - g(s_k)]$ than schemata.

Parameter a is related to time in the evolution process. Considering that the good schemata need to be preserved for recombination, the *evolution parameter* a starts from 0 , and then increases slowly, in small time intervals, from generation to generation. The population at the evolution time a , denoted by P_a , is dynamic in size according to the value of the adaptive parameter a , and can be emptied during the process.

The parameter a is now isolated in expression *(2.4.1)* , thus yielding the following expression and corresponding rank to s_i : $a \geq \dfrac{dg_{\max} - [g(s_i) - f(s_i)]}{d[g_{\max} - g(s_i)]} = d(s_i)$.

At the time they are created, structures and/or schemata receive their corresponding rank value $d(s_i)$. The *rank* of each schema or structure is compared with the current evolution parameter a . At the moment a structure or schema is created, it is then possible to have some figure of its survivability. The higher the value of $d(s_i)$, and better is the structure or schema to the *BOP*, and they also have more surviving and recombination time.

2.5. Selection and recombination

The population was kept in a non-decreasing order according to the following key

$\Delta(s_i) = (1+d_i)/(n-n_\#)$, where $d_i = [g(s_i) - f(s_i)]/g(s_i)$, $n_\#$ is the number of # labels in $s_{i.}$ Schemata with small $n_\#$ and/or presenting small d_i are better and appear in first order positions.

The method used for selection takes one schema from the *n* first positions in the population (*base*) and the second schema from the whole population (*guide*). Before recombination, the first schema is complemented to generate a structure representing a feasible solution, i.e. all #'s are replaced by 0's. This complete structure suffers mutation and is compared to the best solution found so far (which is kept throughout the process). The recombination merges information from both selected schemata, but preserves the number of labels 1 (number of colors) in the new generated schema.

Recombination

```
        if $s_{base}(j) = s_{guide}(j)$ then $s_{new}(j)$ ¬ $s_{base}(j)$
        if $s_{guide}(j)=\#$ then $s_{new}(j)$ ¬ $s_{base}(j)$
        if $s_{base}(j) = \#$ or 0 and $s_{guide}(j)=1$ then
            $s_{new}(j)$ ¬ 1 and $s_{new}(i)$ ¬ 0 for some $s_{new}(i)=1$
        if $s_{base}(j) = 1$ and $s_{guide}(j)=0$ then
            $s_{new}(j)$ ¬ 0 and $s_{new}(i)$ ¬ 1 for some $s_{new}(i)=0$
```

At each generation, exactly *n* new schemata are created by recombination. If a new schema does not represent a feasible solution, then it is inserted into the population; otherwise it suffers mutation and is compared to the best solution found so far. The following pseudo-code describes the mutation process:

Mutation Process

```
1:    For each cluster
            Move the seed to the vertex with the 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.
```

2.6. The algorithm

The Constructive Genetic Algorithm can be summed up by the pseudo-code:

CGA

```
Given $g_{max}$ and $d$ ;
$\alpha := 0$ ;
$\varepsilon := 0.05$;                                   { time interval }
Initialize $P_\alpha$ ;                                  { initial population }
Evaluate $P_\alpha$ ;                                    { fg-fitness }
For all $s_i$ Î $P_a$ compute $d(s_i)$                   { rank computation }
end_for
While (not stop condition) do
    For all $s_i$ Î $P_a$ satisfying $\alpha < d(s_i)$ do   { evolution test }
        $\alpha := \alpha + \varepsilon$ ;
        Select $P_\alpha$ from $P_{\alpha-\varepsilon}$ ;          { reproduction operator }
        Recombine $P_\alpha$ ;                           { recombination operators }
        Evaluate P_α ;                          { fg-fitness }
    end_for
    For all new $s_i$ Î $P_a$ compute $d(s_i)$      { rank computation }
    end_for
end_while
```

3. The column generation

The column generation process was proposed by Mehrotra and Trick (1995). The master problem (*MP*) is

$$\text{Min} \sum_{j\in J} x_j$$

$$\text{Subject to} \sum_{j\in J} x_{ij} \geq 1,\ i=1,\dots,|V|$$

$$x_{ij} \in \{0,1\},\ i=1,\dots,|V|;\ j\in J.$$

Where J is the set of all maximal independent sets of G. It is a set covering problem with a large number of (generally) unknown columns, that are generated when necessary, solving the following weighted maximum independent set problem (*WMIP*)

$$\text{Max} \sum_{i\in V} l_i z_i$$

$$\text{Subject to} \quad z_i + z_j \leq 1, \forall (i,j)\in E$$

$$z_i \in \{0,1),\ \forall i \in V.$$

Where l_i are dual variables for each constraint in *MP*. An initial pool of columns must be given to form the initial *MP*, and columns are elected to enter the *MP* if they return bounds larger than 1 when solving *WMIP*.

The *CGA* described in section 2 was used here to form the initial pool of columns to *MP*. Initially, it is set a number of colors, and if the *CGA* find this specified coloring, this number is reduced, until no more improvement is found. A number of independent sets found during the last *CGA* application is stored to compose the first pool of columns. The optimal solution of *MP* may give a lower bound to the coloring problem, and the dual variables are saved to be used on problem *WMIP*.

In the sequence, the same *CGA* is used to approximately solve problem *WMIP*, setting the last used number of colors minus one, and storing the independent sets found. These new independent sets are appended to the previous pool of columns and problem *MP* is resolved. The process continues until no more columns are found to be added to the *MP*.

A lower bound to the coloring problem is obtained at each iteration applying the Farley´s bound (Farley, (1990)), given by $v(MP)/v(WMIP)$, where v(.) is the optimal value of the corresponding problem. These values change at each process iteration.

4. Computational tests

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 1* bellow gives computational results. The table contains average numbers of vertic es, 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*	*RLF CGA*
Books	4	94.7	363.5	0
Games	1	120	638	0
Miles	3	128	1223.3	0
Register	4	204.8	3848.6	0
Mycielski	3	111	1117	0
Queen	5	51	753.2	0.5

Table1: CGA computational results

The quality of the results can be easily seen, especially for the *queen graphs*, considered hard. Problem Queen99 was the only one for whic h the chromatic number was not reached by the *CGA*.

Then we have proceeded with tests for column generation using the Queen99 instance. Table 2 shows the results.

Process iteration	*Number of colors*	*Best CGA number of conflicts*	*MP bound*	*Farley´s Bound*	*Time (sec.)*
0	10	2	9.226	8.359	295
1	9	10	9.059	8.155	283
2	8	25	9.007	8.542	246
3	7	47	9.000	-	177
4	6	75	-	-	121

Table2: Column generation process for Queen99

The times in *table 2* correspond to the *CGA* application. The *CPLEX 6.5* solves the *MP* problem very fast, and their times are not reported. The best solution found by the *CGA* was 11 colors. Considering the best *MP* and Farley´s bounds, it can be conjectured that the best number of colors to Queen99 is 9, 10 or 11 (actually the best number is 10). As the number of colors decreases, increases the number of conflicts in *CGA* solutions, but the new columns improve the *MP* bounds. The Farley´s bound have an oscillating behavior due to the fact that problem *WMIP* was not exactly solved. We have used a SUN-ULTRA30 and the CGA parameters was iteration_limit=20, *a*_increase=0.01, *d*=0.15 and *mult*=2.0.

To complement this work, other tests must be done with larger graphs and also the *CGA* parameters must be analyzed.

Acknowledgments: The second author acknowledges Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (proc. 350034/91-5, 520844/96-3, 680082/95-6) and Fundação para o Amparo a Pesquisa no Estado de S. Paulo - FAPESP (proc. 95/9522-0 e 96/04585-6) for partial financial support.

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.

Farley, A A (1990) A note on bounding a class of linear programming problems, including cutting stock problems. *Operations \research* 38(5): 922-923.

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

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 algorit hm for the set partitioning problem. In *Proceedings of the 5th International Conference on Genetic Algorithms* , pp. 481-487.

Lorena, L.A.N. and Furtado, J.C. *Constructive genetic algorithm for clustering problems.* Submitted for publication – Evolutionary Computation. Presented at the Optimization 98 congress - Coimbra, Portugal - July 1998. Available from http://www.lac.inpe.br/~lorena/cga/cga_clus.PDF

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.

Mehrotra, A and Trick, M. A (1995) A column generation approach for graph coloring. Available from http://mat.gsia.cmu.edu/color.ps

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-Verla g, *Proceedings 1st International Workshop on Parallel Problem Solving from Nature, Lecture Notes in Computer Science* 496: 109-116.