

# A New Genetic Graph Coloring Heuristic

Cornelius Croitoru, Henri Luchian, Ovidiu Gheorghies, and  
Adriana Apetrei

University “Al. I. Cuza” Iasi, Romania  
Faculty of Computer Science  
{croitoru,hluchian,ogh,adrianaa}@infoiasi.ro

**Abstract.** The aim of this paper is to introduce a new evolutionary formulation of the graph coloring problem, based on the interplay between orderings and colorings of vertices. The new formulation breaks symmetry in the solution space and provides opportunities for combining evolutionary and other search techniques. Our formulation is very simple compared to previous approaches which use the relationship between a graph’s chromatic number and its acyclic orientations. We outline an initial version of a Genetic Algorithm which implements this approach to the coloring problem. We summarise the results obtained from a set of systematic experiments on DIMACS Computational Challenge graphs; they provide convincing evidence that the new ordering approach can lead to accurate genetic coloring solvers, despite the contrary popular belief.

## 1 Introduction

If  $G = (V, E)$  is a graph and  $k$  a positive integer, a  $k$ -coloring of (the vertices of)  $G$  is any assignment  $c : V \rightarrow \{1, \dots, k\}$  with the property that for each  $i \in \{1, \dots, k\}$  the set  $c^{-1} = \{v | v \in V, c(v) = i\}$  is a stable set in  $G$ , that is a set of mutually non-neighbor vertices. The set  $c^{-1}(i)$  is called the color class  $i$  of the coloring  $c$  and the list  $(S_1, \dots, S_k)$  of all color classes completely determines the coloring  $c$ .

It is wellknown that it is NP-complete to decide whether, for a given graph  $G$  and an integer  $k$  there exists a  $k$ -coloring of  $G$  [1], [2]. The least possible number  $k$  of colors for which a graph  $G$  has a  $k$ -coloring is called the chromatic number of  $G$  and is denoted by  $\chi(G)$ .

The graph coloring problem is the problem of finding, for a given graph  $G$ , an optimal coloring, that is a coloring with  $\chi(G)$  colors. Since this problem is not approximable within  $|G|^{1/14-\epsilon}$  for any  $\epsilon > 0$  [3], the graph coloring problem is one of the favorites to try new meta-heuristics for (e.g. simulated annealing [4], tabu search [5]). Other approaches include DSatur [6] and greedy strategies which have been outperformed by hybrid techniques based on maximal stable sets [7], [8].

The aim of this paper is to provide a new evolutionary formulation of the graph coloring problem which is based on the interplay between orderings and

colorings of the vertices. An important consequence of this interplay is that it breaks the symmetry in the solution space. In the same time, the new formulation gives the possibility of combining evolutionary and other search techniques. We note that our formulation is very simple compared with that given in [9] which uses the wellknown relationship that exists between a graph's chromatic number and its acyclic orientations [10].

## 2 Orderings and Colorings

Let  $G = (V, E)$  be a graph with the vertices set  $V = \{1, 2, \dots, n\}$ . The set of all  $n!$  permutations on the set  $V$  is denoted by  $\mathcal{S}_n$  and each permutation  $v \in \mathcal{S}_n$ ,  $v = v_1v_2\dots v_n$ , is interpreted as an ordering of vertices of  $G$ :  $v_1$  is the first vertex in the ordering  $v$ ,  $v_2$  is the second one and so on.

For  $v \in \mathcal{S}_n$  and  $i, j \in \{1, \dots, n\}$ ,  $i < j$  we denote by  $I[v, i, j]$  the interval determined by the two indices  $i$  and  $j$  in  $v$ , that is,  $I[v, i, j] = \{v_{i+1}, \dots, v_{j-1}\}$ . Clearly,  $I[v, i, i+1] = \phi$  for every ordering  $v$  and  $i \in \{1, \dots, n-1\}$ .

**Definition 1.** *Let  $e \in E$  be an edge of  $G$  and  $v \in \mathcal{S}_n$  an ordering. We call  $e$  a bad edge with respect to  $v$  if  $e = v_iv_j$ ,  $i < j$  and  $I[v, i, j]$  is a stable set in  $G$ .*

Note that by this definition any edge connecting two consecutive vertices in  $v$ ,  $e = v_iv_{i+1}$ , is a bad edge with respect to  $v$  because  $I[v, i, i+1] = \phi$  is a trivial stable set.

For each ordering  $v \in \mathcal{S}_n$  we denote by  $b(v)$  the number of all bad edges in  $G$  with respect to  $v$ .

The following theorem and its proof show that determining an optimal coloring in a graph is equivalent to finding an ordering of its vertices with a minimum number of bad edges.

**Theorem 1.**

$$\chi(G) = 1 + \min_{v \in \mathcal{S}_n} b(v)$$

*Proof.* 1. Let  $v^0 \in \mathcal{S}_n$  be an ordering with minimum number of bad edges. Starting with  $v_1^0$  construct a maximal stable set of consecutive vertices in  $v^0$ :  $\{v_1^0, \dots, v_i^0\}$ . Clearly,  $i \geq 1$  and either  $i+1 > n$  or  $v_{i+1}^0$  is adjacent with at least one vertex from  $\{v_1^0, \dots, v_i^0\}$ . Call this stable set  $S_1$  and repeat the above construction, starting with  $v_{i+1}^0$  and obtaining  $S_2, S_3, \dots$ , until all vertices of  $v^0$  have been considered. We obtain a  $k$ -coloring  $(S_1, S_2, \dots, S_k)$  of  $G$  with the property that for each  $t \in 2, \dots, k$ , if  $v_j^0$  is the first vertex (with respect to the ordering  $v^0$ ) of  $S_t$ , then there is a last vertex  $v_i^0$  in  $S_{t-1}$  such that  $v_i^0v_j^0 \in E$ . Clearly, by the choice of  $v_i^0$ , the edge  $v_i^0v_j^0$  is a bad edge with respect to  $v^0$ . Thus,

the number  $k$  of colors in the coloring associated with  $v^0$  satisfies the inequality  $k - 1 \leq b(v^0)$ . Since  $k \geq \chi(G)$ , we have

$$\chi(G) \leq k \leq 1 + b(v^0) \tag{1}$$

2. Let  $(S_1, S_2, \dots, S_k)$  be an optimal ( $k = \chi(G)$ ) coloring of  $G$ . Each stable set  $S_i$ ,  $1 \leq i \leq k-1$  can be extended to become a maximal stable set in the subgraph induced in  $G$  by  $S_i \cup S_{i+1} \cup \dots \cup S_k$ . The number of stable sets obtained remains  $k$  since  $k = \chi(G)$ . The obtained  $k$ -coloring  $(S_1, S_2, \dots, S_k)$  has the property that each vertex  $u \in S_i$  has at least one neighbour in  $S_{i-1}$ ,  $2 \leq i \leq k$ . Using this property we construct an associated ordering  $v^0 \in \mathcal{S}_n$  in the following way:

- (a) the last vertices in the ordering  $v^0$  are the vertices of  $S_k$  in an arbitrary order; if  $S_k = \{s_1, \dots, s_p\}$  then  $v_n^0 = s_1, v_{n-1}^0 = s_2, \dots, v_{n-p+1}^0 = s_p$ ;
- (b) the vertex before  $v_{n-p+1}^0$  (i.e.  $v_{n-p}^0$ ) is a vertex  $u \in S_{k-1}$  which is a neighbour of  $v_{n-p+1}^0$ ; there is such a vertex by the above property of the coloring. Put  $v_{n-p} = u$  and the remaining vertices of  $S_{k-1}$  are added to  $v^0$  in an arbitrary order (i.e.  $S_{k-1} - \{u\} = \{u_1, \dots, u_t\}$ , put  $v_{n-p-1}^0 = u_1, v_{n-p-2}^0 = u_2, \dots, v_{n-p-t}^0 = u_t$ );
- (c) repeat (b) for  $S_{k-2}, S_{k-3}, \dots$  until all vertices of  $S_1$  (and therefore of  $G$ ) are added to  $v^0$ .

The ordering  $v^0$  constructed above has only  $k - 1$  bad edges, namely those obtained in the step (b) of the algorithm, connecting consecutive vertices in  $v^0$  which have also consecutive colors. Therefore we have  $b(v^0) + 1 = k = \chi(G)$  and since  $b(v^0) \geq \min_{v \in \mathcal{S}_n} b(v)$  we obtain

$$\chi(G) = b(v^0) + 1 \geq 1 + \min_{v \in \mathcal{S}_n} b(v) \tag{2}$$

By (1) and (2) the theorem holds. □

Let us note that by (1) and (2) above the coloring and the ordering constructed in the proof are optimal. In other words, the two constructions described in the proof show how to efficiently obtain an optimal coloring from an ordering with minimum number of bad edges and conversely.

### 3 Description of the Genetic Algorithm

This section describes a genetic algorithm based on the ordering approach of coloring.

Genetic algorithms [11] are probabilistic techniques which mimic the natural evolutionary process. A genetic algorithm maintains a population of candidate solutions. This is one important feature that differentiates them from conventional heuristics like hill climbing, simulated annealing, tabu search etc. Genetic algorithms have thus the potential to better explore the search space; moreover, they provide tools for avoiding “premature convergence”.

Each candidate solution in the population is encoded into a structure called chromosome. To each chromosome, a value (called fitness value) is assigned. The fitness value represents the quality of the candidate solution encoded by the chromosome. Assigning fitness values to chromosomes is called evaluation.

A selection procedure simulates the “survival of the fittest” paradigm from nature. Better-fitted chromosomes have higher chances of surviving to the next generation. The number of chromosomes per generation is constant.

As in natural life, offspring chromosomes are obtained from parent chromosomes. One possibility is for two parents to exchange encoded information and thus creating two new offspring; this is called crossover. Another possibility is to alter the encoded information in a chromosome obtaining a slightly different new chromosome; this is called mutation. Some other chromosomes simply survive unaltered, while others die off. Mutation and crossover are referred to as genetic operators.

### 3.1 Representation

A chromosome represents an ordering of the vertices of the graph. We encode the ordering by means of a permutation.

If the graph has  $n$  vertices, the chromosome will be a vector

$$chrom = (v_1, v_2, \dots, v_n), \text{ where } v_i \in \{1, 2, \dots, n\}, v_i \neq v_j, i \neq j \quad (3)$$

The candidate solution represented by the chromosome is obtained in a straightforward manner: the position of vertex  $v_i$  in the ordering is  $i$ .

### 3.2 Genetic Operators

We define one crossover operator and four mutation operators. Each operator has a rate, which results in the number of chromosomes to which that operator is applied. These rates are parameters of the genetic algorithm; they are tuned in order to balance exploration and exploitation [11].

**Crossover** The crossover is designed to propagate and exchange information regarding stable sets defined in the parents throughout evolution. Since stable sets are supposedly found in consecutive nodes in parent chromosomes, there are blocks of nodes which are inherited by offspring. There is no a priori reason for such blocks to have the same size, which leads to the idea of using two possibly different cutting points for each parent.

Let  $parent^1 = (v_1^1, v_2^1, \dots, v_n^1)$  and  $parent^2 = (v_1^2, v_2^2, \dots, v_n^2)$  be two parent chromosomes. We consider two randomly generated cutting points  $c^1, c^2 \in \{1, 2, \dots, n - 1\}$  for  $parent^1$  and  $parent^2$  respectively.

One offspring is obtained by keeping unaltered the genetic information from  $parent^1$  before  $c^1$ ; the vertices after  $c^1$  from the first parent are rearranged using the ordering defined by the second parent. The second offspring is constructed in a similar way.

**Example.** Suppose we have to color a graph  $G = (V, E)$ , where  $|V| = 6$ . Let us consider  $parent^1$  and  $parent^2$  two parent chromosomes, and the cutpoints  $c^1 = 3$  and  $c^2 = 2$ . The corresponding  $offspring^1$  and  $offspring^2$  are depicted below.

$$\begin{array}{cc} parent^1 316^\downarrow 254 & offspring^1 316524 \\ \Rightarrow & \\ parent^2 52^\downarrow 4136 & offspring^2 523164 \end{array}$$

**Order Mutation (OM)** Any mutation operator is aimed at sampling the search space in a neighborhood of a chromosome. For order mutation this is achieved by changing the order of some (few) vertices.

Let  $parent = (v_1, v_2, \dots, v_n)$  be a parent chromosome. The offspring differs from the parent by a randomly generated number  $l$  of interchanges between vertices. The interval for the values of  $l$  is empirically adjusted before the genetic algorithm is run. Small values for  $l$  may lead to a too focused exploration, while large values may alter dramatically the genetic heritage. The interval is scaled to the number of vertices in the graph. This way, the dependence between the performance of the genetic algorithm and the instance of the problem is diminished.

**Example.** For a graph with 6 vertices, let  $parent$  be a parent chromosome. A mutation with  $l = 1$  produces the  $offspring$  depicted below.

$$parent 31 \overset{\downarrow}{6} 25 \overset{\downarrow}{4} \Rightarrow offspring 314256$$

**Block Mutation (BM)** This type of mutation performs translations of blocks of  $k$  successive vertices ( $k$  plays a role similar to  $l$  above).

Let  $parent = (v_1, v_2, \dots, v_n)$  be a parent chromosome. If  $k = 2$  and  $i \in [1, n - 1], j \in [1, n]$  are randomly generated, then block mutation yields (for  $j > i$ ):

$$offspring = (v_1, \dots, v_{i-1}, v_{i+2}, \dots, v_j, v_i, v_{i+1}, v_{j+1}, \dots).$$

**Color Spread Mutation (CSM)** Let  $parent = (v_1, v_2, \dots, v_n)$  be a parent chromosome and  $e = v_i v_j$  a randomly picked bad edge and  $e' = v_k v_l$  the next bad edge ( $i < j < k < l$ ). In the CSM offspring chromosome, the nodes from positions  $i + 1$  to  $l - 1$  are moved to randomly chosen new positions in the permutation.

In other words, all nodes having the same color as  $v_j$  are randomly relocated.

**Bad Edge Stretch Mutation (BESM)** This type of mutation attempts at reducing the number of bad edges.

Let  $parent = (v_1, v_2, \dots, v_n)$  be a parent chromosome and  $e = v_i v_j$  a randomly chosen bad edge. A direction (left-to-right or right-to-left) is randomly chosen.

Suppose that the chosen direction is left-to-right (the other case is analogous). Let  $v_k$  be the farthest (in the ordering) node adjacent to  $v_i$  so that  $k > j$ . Let  $e' = v_l v_m$  be the first bad edge so that  $l > k$ . If there is no node  $v_k$  or edge  $e'$  with the specified properties, the mutation is void. Otherwise, the node  $v_i$  is moved to the position  $m + 1$ . This way, the edge  $e = v_i v_j$  will no longer be a bad edge, and the edges  $e'' = v_i v_s$  with  $j < s \leq k$  remain non bad edges. New bad edges may appear. Experiments indicate that the number of bad edges does not increase when BESM is applied.

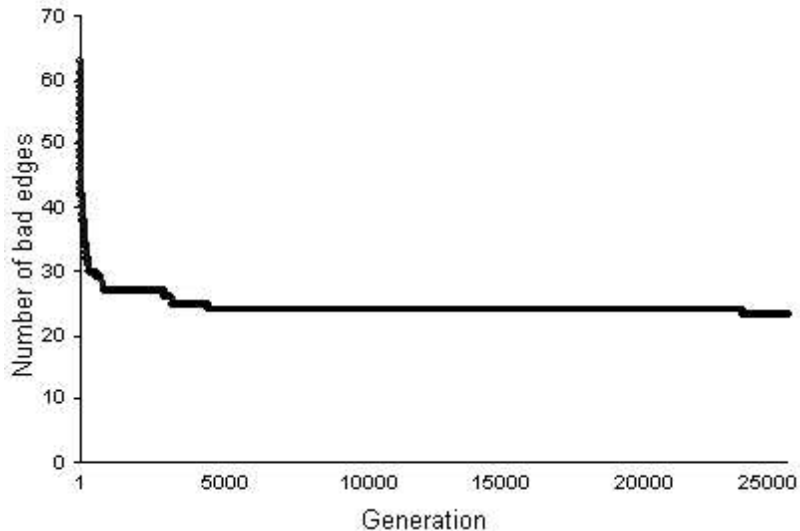
### 3.3 Evaluation

The purpose of evaluation is to point out how good the candidate solution encoded by a chromosome is. In order to provide such a measurement, a coloring should be constructed from each ordering (chromosome). The number of colors of the coloring (or, alternatively, the number of bad edges) is taken as a measure of an individual's quality. This is consistent with the minimization of the number of colors.

**Evaluation by Bad Edges** One approach is to minimize the number of bad edges. The algorithm used for counting the number of bad edges is the one presented in the first part of the proof of theorem 1. According to this algorithm, each stable set contains successive nodes in the ordering. Although a near optimal coloring is often found, this evaluation method makes the genetic algorithm very sensitive to small changes in an ordering, and causes it to converge very slowly (figure 1). Typical results obtained using this evaluation procedure are given in table 1.

**Evaluation by Heuristics** It is possible to increase the convergence speed by using heuristics. The heuristics we use start off with an initial ordering, which is then transformed into another ordering with an eye to obtaining a reduced number of bad edges.

Typically, a heuristic for graph coloring consists of two parts: in the first part, a suitable ordering of the vertices is fixed and in the second part, the actual coloring algorithm is applied using the fixed order of vertices. A very basic coloring procedure is the *sequential algorithm* (sometimes called *greedy algorithm*), which proceeds as follows. Assume the vertices of the graph are given in the order  $v_1, v_2, \dots, v_n$ ; they will be processed in this order. Assign color 1 to  $v_1$ . For each of the remaining vertices  $v_i$ , assign to  $v_i$  the minimum color number available, that is, the smallest color number that, so far, has not been assigned to any vertex adjacent to  $v_i$ . Though the local action of the sequential algorithm appears to be quite reasonable, globally it may fail miserably for unfavorable vertex orderings. It is not difficult to construct a sequence  $G_3, \dots, G_m, \dots$  of graphs such that each  $G_m$  is 2-colorable, the size of  $G_m$  is linear in  $m$ , and yet the number of colors used by the sequential algorithm on input  $G_m$  is at least  $m$  for at least one vertex ordering. Similar results can be obtained for other graph



**Fig. 1.** Evolution of the number of bad edges per generation (DSJC125.5)

coloring heuristics, most of which apply the sequential coloring algorithm to various vertex orderings that are obtained by seemingly reasonable procedures.

Our genetic approach gives, by its probabilistic nature, the possibility to avoid such faulty orderings and, as suggested our computational experiences, the use of the number of bad edges (as a measure of the quality of a particular ordering) allows to escape from local minima and obtain bounds close to the absolute minimum, provided that suitable genetic operators are used.

Furthermore, these computational results conducted us to a novel approach to the above sequential procedure. Clearly, the sequential coloring can be equivalently implemented by finding successive lexicographic-first maximal stable sets with respect to a previously fixed ordering  $v_1, v_2, \dots, v_n$ , until all vertices are considered. If the order of the remaining vertices is changed after each construction of a color class it seems that this adaptive coloring algorithm proceeds better. A good candidate for a dynamic changing of the ordering can be deduced from the following interesting observation.

**Proposition 1.** *Let  $G = (V, E)$  be a graph and  $w_1, w_2, \dots, w_n$  a vertex ordering generated by a breadth first traversal of  $G$ . If  $S$  is the lexicographic-first (with respect to the BFS ordering  $w_1, w_2, \dots, w_n$ ) maximal stable set of  $G$ , then the bipartite graph obtained from  $G$  by deleting all edges with both extremities in  $V - S$ , has the same connected components as the graph  $G$ .*

Indeed, we can suppose that  $G$  is connected and it is not difficult to show that any vertex  $v$  of  $G$  can be reached from the vertex  $w_1$  on a path that uses edges with only one extremity in  $V - S$  (by induction on the distance in  $G$  from

$v$  to  $w_1$ ). The maximal stable set  $S$  has, by the above proposition, the property that it is incident to many edges (at least  $n - 1$ , if  $G$  is connected) in a somewhat uniform way. We note that there are nondeterministic choices to be made whenever there are more neighbors not yet visited at any point in the BF traversal of  $G$ . These ties will be solved by using an initial ordering  $v_1, v_2, \dots, v_n$ . Our new graph coloring heuristic can be described as:

Use a particular ordering  $v_1, v_2, \dots, v_n$  to direct all the BF traversals

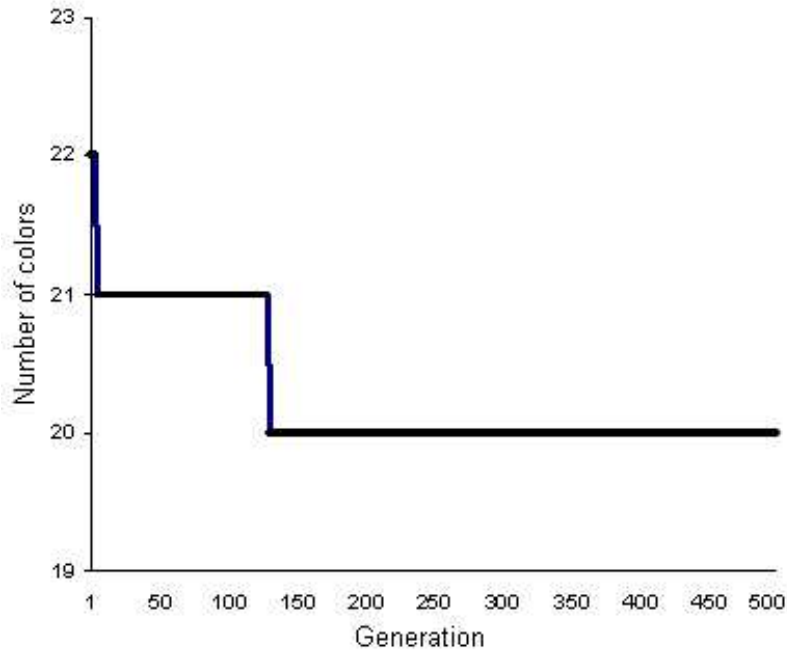
While  $G$  is not empty do

```

  Apply a BF traversal to  $G$  and find the first
  (in the lexicographic BF ordering obtained) maximal stable set  $S$ ;
  Let  $S$  be the new color class;
   $G := G - S$ 

```

Experimental results show that the use of these heuristics has a favorable effect on the convergence speed (figure 2). Moreover, as shown in table 1, the solutions are better than those obtained by simple bad edge counting.



**Fig. 2.** Evolution of the number of colors using heuristics (DSJC125.5)

### 3.4 Selection

The selection we use is rank-based. The chromosomes are ordered according to their fitness values. Chromosomes having the same fitness value are grouped into one rank (there are as many ranks as there are different fitness values). We build a probability field which indicates what are the chances for a particular rank to be selected. Once a rank is selected, a chromosome from that rank is randomly picked.

## 4 Experimental Results

The experiments were carried out using the `even` genetic algorithms library [13].

We have run our genetic algorithm on some of the DIMACS benchmark graphs [12]. Table 1 gives the summary of our experiments. The statistics presented here is the minimum number of colors ( $k$  min) over eight runs of the genetic algorithm for each instance of the problem. We also specify for each graph the number of vertices, the number of edges, and the upper bound for the chromatic number (if known).

The experiments were run for various settings of the genetic algorithm. Most runs used the following settings: order mutation rate around 0.1 with minimal and maximal interchange rate of 0.05 and 0.1 respectively, block mutation move rate around 0.1 with minimal and maximal block size rate of 0.05 and 0.2 respectively, color spread mutation rate around 0.1, crossover rate around 0.5. The probability of selecting the chromosomes from the best rank was set to 0.2.

In figure 3 several typical runs of the bad edge based genetic algorithm for the graph DSJC125.5 are presented. Note that the algorithm converges quite satisfactory using only the crossover [(2) in figure 3]. The order mutation has a destructive effect [(1) in figure 3]. The best results for this graph have been obtained by using a larger population and applying the block mutation and color spread mutation, along with the crossover [(3) and (4) in figure 3]. When the bad edge stretch mutation is used, the convergence is significantly improved in the first part of the run, but then local optimum is reached.

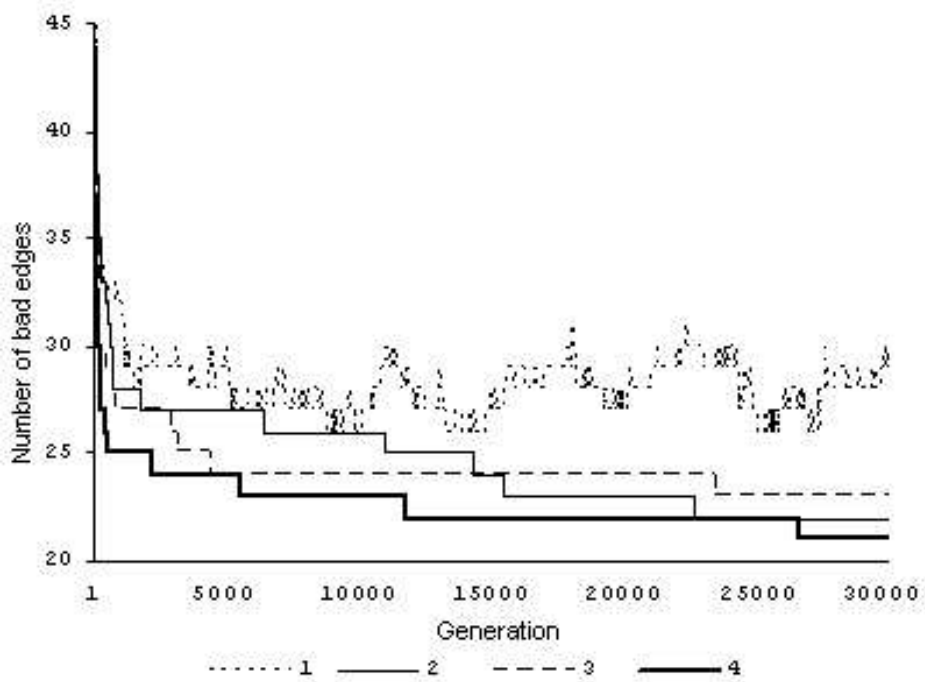
## 5 Conclusions and Future Work

We have presented a novel approach to the graph coloring problem, suitable for evolutionary implementations. An initial version of such an implementation has also been outlined.

The general feeling in the research community is that genetic algorithms do not cope well with the coloring problem. Despite of this, our first sets of systematic experiments provided convincing evidence that the new ordering approach can lead to accurate genetic coloring solvers. In order to achieve this, more hybridisation of the genetic algorithm has to be done: on one hand, hybridisation with the problem (e.g., sequential colorings used for genetic operators) and, on the other hand, hybridisation with other meta-heuristics.

**Table 1.** Experimental results

Graph name	V	E	$\chi(G)$	$k$ min	
				bad edge	heuristics
1-Insertions.4	67	232	$\leq 4$	4	5
1-Insertions.5	202	1227		7	6
1-Insertions.6	607	6337		27	7
2-Insertions.3	37	72	$\leq 4$	4	4
2-Insertions.4	149	541	$\leq 4$	4	5
2-Insertions.5	597	3936		20	6
3-Insertions.3	56	110	$\leq 4$	4	4
3-Insertions.4	281	1046		7	5
3-Insertions.5	1406	9695		40	6
1-FullIns.3	30	100		4	4
1-FullIns.4	93	593		5	5
1-FullIns.5	282	3247		16	6
2-FullIns.3	52	201		5	5
2-FullIns.4	212	1621		10	6
2-FullIns.5	852	12201		44	7
3-FullIns.3	80	346		5	6
3-FullIns.4	405	3524		18	7
3-FullIns.5	2030	33751		94	8
4-FullIns.3	114	541		6	7
4-FullIns.4	690	6650		29	8
4-FullIns.5	4146	77305		158	9
fpsol2.i.1	496	11654	$\leq 65$	66	65
fpsol2.i.2	451	8691	$\leq 30$	36	30
fpsol2.i.3	425	8688	$\leq 30$	34	30
inithx.i.1	864	18707	$\leq 54$	62	54
inithx.i.2	645	13979	$\leq 31$	38	31
inithx.i.3	621	13969	$\leq 31$	37	31
DSJC125.5	125	3891	$\leq 17$	22	20
DSJC250.5	250	15668	$\leq 28$	62	37
DSJC500.1	500	12458		47	16
DSJC500.5	500	62624		128	66
DSJR500.1	500	3555		28	12
DSJR500.1c	500	121275		56	88
le450_15a	450	8168	$\leq 15$	42	18
le450_15b	450	8169	$\leq 15$	42	18
le450_15c	450	16680	$\leq 15$	58	27



**Fig. 3.** Typical runs of the genetic algorithm for the graph DSJC125.5 (30000 generations). (1) Population size 100, OM rate 0.15 bounds (0.01, 0.1), BM rate 0, CSM rate 0, crossover rate 0.6. (2) Population size 100, OM rate 0, BM rate 0, CSM rate 0, crossover rate 0.6. (3) Population size 1000, OM rate 0.1 bounds (0.01, 0.1), BM rate 0.1 bounds (0.01, 0.2), CSM rate 0.1, crossover rate 0.5. (4) Population size 1000, OM rate 0, BM rate 0.15 bounds (0.01, 0.2), CSM rate 0.15, crossover rate 0.6.

## References

1. V.C. Barbosa, C.A.G. Assis and J.O. do Nascimento *Two Novel Evolutionary Formulations of the Graph Coloring Problem*, available electronically.
2. M. Bellare and M. Sudan, *Improved non-approximability results*, Proceedings 26th Ann. ACM Symposium on Theory of Computation, ACM, 1994, 184-193.
3. D. Brélaz, *New methods to color vertices of a graph*, Communications of the ACM 22, 1979, 251-256.
4. J. Culberson and F. Luo, *Exploring the k-Colorable Landscape with Iterated Greedy*, Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge, Providence, RI: AMS, 1996, 245-284.
5. R.W. Deming, *Acyclic orientations of a graph and chromatic and independence numbers*, Journal of Combinatorial Theory B 26, 1979, 101-110.
6. P. Galinier and J.K. Hao, *Hybrid evolutionary algorithms for graph coloring*, Journal of Combinatorics 3(4), 1999, 379-397.
7. M.R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, NY, 1979.
8. A. Hertz and D. de Werra, *Using tabu search techniques for graph coloring*, Computing 39(4), 1987, 345-351.
9. D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, *Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and member partitioning*, Operations Research 39 (1991), 378-406.
10. R.M. Karp, *Reducibility among combinatorial problems*, in R.E. Miller and J.W. Thatcher (Eds.), *Complexity of Computer Computations*, Plenum Press, New York, NY, 1972, 85-103.
11. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3<sup>rd</sup> edition, Springer Verlag, 1996.
12. DIMACS Benchmark Graphs, <http://mat.gsia.cmu.edu/COLORING02/index.html>
13. Ovidiu Gheorghies, *even*, *A C++ Genetic Algorithms Library*, available electronically at <http://students.infoiasi.ro/~cevol>.