Greedy algorithms for dynamic graph coloring

Linda Ouerfelli
LARODEC Laboratory, ISG,
University of Tunis, Tunisia
Email: linda.ouerfelli@ymail.com

Hend Bouziri
LARODEC Laboratory, ESSEC,
University of Tunis, Tunisia
Email: hend.bouziri@gnet.tn

***Abstract*—Many real life applications are subject to changes which can be modeled as dynamic graphs. In this paper, we are interested especially in the dynamic graph coloring. We focus on coloring using online algorithms and we propose new greedy approaches to solve it efficiently.**

I. Introduction

The graph coloring problem is a classical problem of combinatorial optimization. It is applied in several real life areas such as timetabling, scheduling, frequency assignment and register allocation. Other specific applications can be found in the literature for example air traffic management and train platforming [1] [2]. For these applications, we have to assign a limited number of resources (colors) to a set of variables (nodes) under incompatibility constraints (edges).

Too many work was proposed to deal with the graph coloring problem. These methods arise from exact techniques to meta heuristics [3] [4] [5]. In all these techniques the graph to be colored is static, i.e the sets of nodes and edges are given in advance.

However, in real life applications variables and incompatibility constraints can evolve with time. Thus, we have to deal with dynamic graphs. Few work has been devoted to deal with the dynamic graph coloring problem [6] [7]. The main drawback of these techniques is that they are efficient for specific dynamic graph instances and they show bad performance with other graphs.

The graph coloring is a well studied problem and known to be an NP-complete problem except for some specific cases such bipartite graphs or trees [8].

In this work we propose new greedy methods and we perform many experiments to show the efficiency of our approaches. In the next section, we present the classical graph coloring problem. In section III, we deal with the definition of dynamic graphs. Then the problem of online graph coloring is detailed in section IV. Our new methods are presented in section V. Finally, experiments are analyzed in section VI.

II. Graph coloring problem

In this section, we present the classical problem of graph coloring with some definitions and algorithms proposed to solve it.

A. Definitions

A graph $G = (V, E)$ is defined by a non-empty set of vertices V and a set of edges E. Two vertices v_i and v_j are called *adjacent* if and only if $(v_i, v_j) \in E$. An *independent set* contains no adjacent vertices.

A proper coloring of the graph G is simply assigning a color to each vertex in such way that two adjacent vertices do not have the same color, as in definition 1. Two adjacent vertices v_i and v_j are said *in conflict* if the colors c_i and c_j assigned to them are the same ($c_i = c_j$).

Definition 1. *A coloring of* $G = (V, E)$ *is a map* $C : V \rightarrow \{C_1, C_2, ..., C_n\}$
A proper coloring of $G = (V, E)$*:* $C(v_i) \neq C(v_j)$ $\forall (v_i, v_j) \in E$

When the number of colors used to solve the problem is the lowest possible it is called the *chromatic number* of G, denoted $\chi(G)$. Finding the chromatic number of a given graph G is known to be NP-hard.

B. Algorithms

Many algorithms were proposed in the literature. Some are exact methods and others are heuristics. In this work we are interested in greedy algorithms. In fact, this type of methods have shown efficiency on several instances of graphs. These techniques for coloring selects the vertex to be colored according to a specific ordering of vertices.

Algorithm 1 The DSATUR algorithm

```
1:  color v with the maximal degree
2:  repeat
3:      color v having the maximal saturation degree
4:      if tie of saturation degrees then
5:          color v having the maximal uncolored neighbors
6:          if tie of uncolored neighbors then
7:              color v randomly
8:          end if
9:      end if
10: until V is empty
```

For example the *DSATUR algorithm* proposed by Brélaz [5], is a sequential algorithm that treats the vertices one by one and establishes a dynamic ordering of the vertices according to the

saturation degree. This degree is the number of different colors of the adjacent vertices of a given vertex. This algorithm will be later used in the experimental section.

III. Dynamic graphs

Mainly, combinatorial optimization problems deal with static data. However, in most real life applications the environment evolves and changes. Indeed, if the treated problem is modeled as a graph we have to take into account the dynamic aspect of the problem.

The dynamic graphs find applications in several domains as biology, chemistry, neuro-science, engineering, transport and computer science [9].

In this work, we consider dynamic graphs as defined in [12].

Definition 2. *Let a dynamic graph $G(t) = (V(t), E(t))$ where $V(t)$ and $E(t)$ are the sets of vertices of edges at time t and the sequence $U = (u_1, u_2, ..., u_n)$ of update events (addition or suppression). So, applying a change u_i on a graph $G(i-1)$ results on the graph $G(i)$ at $t = i$, $G(i) = G(i-1) + u_i$*

IV. Online graph coloring

In this work, we are interested in the online graph coloring. The nodes are given one by one (with their corresponding edges), as in [11] and a color is assigned to the current vertex before the next vertices are presented. Once a color is assigned to a node, changes are not allowed. Also, the number of colors used have to be the lowest possible and all adjacent vertices must have different colors.

In the literature, dynamic graphs are handled in an online manner such as [13] [14] [15] to deal with real applications such as market graph, citation network and interactions network.

In fact, an online algorithm present a suitable framework for this type of graphs since it considers as input a sequence $\sigma = (\sigma_1, \sigma_2, ..., \sigma_n)$ and treats the requests one at a time with a total ignorance of the next requests. Also, online algorithms correspond more to real world applications and are more adequate for a missing informations analysis [16].

This problem can be seen as a game between two players where the first player P_1 is the algorithm and the second P_2 is the adversary that gives the input sequence and tries to force P_1 to use the maximum number of resources. There are three types of adversary:

- *Oblivious*: It prepares the request sequence in advance and ignores the algorithm actions.
- *Adaptative on-line*: It generates the next request according to the previous responses of the algorithm and serves it as soon as it makes the request.
- *Adaptative off-line*: It produces the requests one by one based on the algorithm answers and serves the whole sequence optimally at the end. Dealing with this type of adversary is much more difficult then the others.

The *First-Fit* method is well used as an online algorithm to deal with dynamic graph coloring. It is an intuitive greedy algorithm. It responds to the i^{th} request with the least possible color.

Algorithm 2 The First-Fit algorithm

```
for all (v_i ∈ σ) do
    choose a legal color c_i
    color v ∈ V with c_i
end for
```

The main drawback of *First-Fit* algorithm is its bad performance when dealing with some types of graphs. This bad performance can be explained by the sensibility of the *First-Fit* choice to the input sequence. For example, if we consider a tree with $n = 5$ and an input sequence $\sigma = (a, d, e, c, b)$. Despite the fact that this graph can be colored using only two colors $\chi(G) = 2$, the *First-Fit* uses three colors $k = 3$, as modeled in figure 1.

Fig. 1. A tree colored with 3 colors using the *First-Fit* algorithm

Another online method is the *randomized algorithm* proposed by Vishwanathan [6]. Indeed, randomization makes it harder for the adversary to predict the moves of the algorithm. This algorithm defines a subset of colors C_s and tries to color the current vertex with a color from it. If this fails the vertex is then putted into a residual set R_q where for every vertex $v_i \in R_q$ there is at least a vertex adjacent to it belonging to a color class from C_s. Other way, the vertex can be colored. Then the algorithm tries to color the residual set using a recursive procedure.

V. Greedy algorithms for online coloring

A. The generic greedy algorithm

In this section, we present our contribution in the attempt to solve this problem. We introduce new greedy algorithms. Since the criterion of choice of the colors can not be applied on vertices as in the classical graph coloring, we choose to apply this rule on the colors. Let C_i be the set of nodes colored with c_i and n_i the number of vertices in C_i. This number is incremented each time c_i is used.

We present below the generic greedy algorithm for online coloring.

Algorithm 3 The generic greedy algorithm

```
1: Input: SEQ = sequence of vertices
2: for all (v_i ∈ SEQ) do
3:     repeat
4:         P := generate the list of possible colors
5:         if sizeP = 0 then
6:             move to next color
7:         else
8:             c(v_i) := selection-rule(P)
9:         end if
10:    until v_i is colored
11: end for
```

Given a sequence of vertices (with their incident edges), the algorithm starts by the first node in the sequence SEQ and color it using one from a list of possible colors P according to a specific selection rule.

We propose the use one of this three selection rules:

1) MinG: select the color c_i with the least number of vertices (minimum n_i).
2) MaxG: select the color c_i with the maximum number of vertices (maximum n_i).
3) RandG: select randomly a color.

If the current node is in conflict and can not be colored using the available set of colors then we have to add a new color.

VI. Experiments

In this section, we present the tested instances and we report the experimental results of our greedy algorithms. Also we compare the effectiveness of our approach with the First-Fit algorithm.

A. Experimental setup

Since the existing benchmarks concern static graphs, the input sequence SEQ is created artificially. In fact the sequence is generated by a random choice over the set of vertices. All the algorithms are executed on the same sequence so we can compare the results. We use the graph instances from the ROIS benchmark [17] to perform our experiments.

The types of graphs used are:

- book graphs: includes anna, david, homer, huck and jean where the number of nodes is between 74 and 561.
- fpsol, inithx, mulsol and zerion: generated from register allocation problems and the number of nodes is between 184 and 864.
- myciel, fullins and insertion: triangle free graphs obtained from a Mycielski transformation and the number of nodes is between 11 and 191. Full insertions and k-insertions graphs are a generalization of myciel graphs with an increased size and the number of nodes is between 30 and 4146.
- le450-x: are Leighton graphs where 450 is the nodes number and "x" indicates the chromatic number and the connectivity of the graph.
- DSJCx, DSJRx and DSJRxc: random graphs DSJC with a number of nodes, indicated by the "x" in the name, between 125 and 1000 and geometric graphs DSJR with "c" in the name indicates the complement.

B. The results

To measure the performance of the proposed algorithms, we compare the results with the solution given by the First-Fit algorithm (FF) and with the chromatic number of the tested graph.

TABLE I
Results for book graphs

Graphs	χ	FF	MinG	MaxG	RandG
anna	11	11	11	11	11
david	11	11	11	11	11
homer	13	13	14	13	14
huck	11	11	11	11	11
jean	10	10	10	10	10

In table I, we give the results of the tested algorithm on the book graphs and in table II the results tested on the graphs of the register allocation problem.

The proposed algorithms can find the chromatic number for almost all the instances. The MaxG variant can find the chromatic number for all the instances and it is as competitive as the First-Fit algorithm.

TABLE II
Results for register allocation graphs

Graphs	χ	FF	MinG	MaxG	RandG
fpsol2.i.1	65	65	65	65	65
fpsol2.i.2	30	30	34	30	33
fpsol2.i.3	30	30	33	30	33
inithx.i.1	54	54	57	54	54
inithx.i.2	31	31	38	31	34
inithx.i.3	31	31	38	31	34
mulsol.i.1	49	49	49	49	49
mulsol.i.2	31	31	34	31	32
mulsol.i.3	31	31	35	31	33
mulsol.i.4	31	31	35	31	33
mulsol.i.5	31	31	35	31	33
zeroin.i.1	49	49	49	49	49
zeroin.i.2	30	30	30	30	30
zeroin.i.3	30	30	31	30	30

In table III, we give the results of the tested algorithms on the Myciel graphs. We compare the results of the proposed methods with the results of the First-Fit algorithm and the best known static coloring k^*. For almost all the instances, MaxG can find k^* and it gives also the same results for all the instances as FF method.

TABLE III
RESULTS FOR MYCIEL GRAPHS

Graphs	$\chi\|LB$	k^*	FF	MinG	MaxG	RandG
1-FullIns-3	4	4	4	4	4	4
1-FullIns-4	4	5	5	6	5	5
1-FullIns-5	6	6	6	10	6	9
1-Insertions-4	4	5	5	5	5	5
1-Insertions-5	3	6	6	6	6	7
1-Insertions-6	3	7	7	12	7	11
2-FullIns-3	5	5	5	5	5	5
2-FullIns-4	6	6	6	8	6	7
2-FullIns-5	7	7	8	15	8	13
2-Insertions-3	4	4	4	4	4	4
2-Insertions-4	4	5	5	5	5	5
2-Insertions-5	3	6	7	9	7	8
3-FullIns-3	6	6	6	6	6	6
3-FullIns-4	7	7	7	10	7	10
4-FullIns-3	7	7	7	7	7	7
4-FullIns-4	8	8	8	13	8	10
4-FullIns-5	6	9	11	28	11	22
4-Insertions-3	4	4	4	4	4	4
4-Insertions-4	5	5	5	7	5	6
5-FullIns-3	8	8	8	8	8	8
5-FullIns-4	8	9	9	15	9	13
myciel3	4	4	4	4	4	4
myciel4	5	5	5	5	5	5
myciel5	6	6	6	6	6	6
myciel6	7	7	7	7	7	7
myciel7	8	8	8	10	8	10

TABLE IV
RESULTS FOR LEIGHTON GRAPHS

Graphs	χ	k^*	DSATUR	FF	MinG	MaxG	RandG
le450-5a	5	5	10	12	15	12	15
le450-5b	5	5	9	12	15	12	14
le450-5c	5	5	6	14	19	14	19
le450-5d	5	5	11	15	20	15	19
le450-15a	15	15	16	20	25	20	24
le450-15b	15	15	16	20	24	20	24
le450-15c	15	15	24	29	35	29	34
le450-15d	15	15	24	29	35	28	34
le450-25a	25	25	25	26	30	26	29
le450-25b	25	25	25	26	29	26	29
le450-25c	25	25	29	34	40	34	40
le450-25d	25	25	28	34	41	34	39

TABLE V
RESULTS FOR DSJ GRAPHS

Graphs	$\chi\|LB$	k^*	DSATUR	FF	MinG	MaxG	RandG
DSJC125.1	5	5	6	7	8	7	8
DSJC125.5	12	17	21	23	24	24	25
DSJC125.9	42	44	50	52	54	53	54
DSJC250.1	5	8	10	12	14	12	13
DSJC250.5	14	28	38	40	46	40	44
DSJC250.9	48	72	91	93	98	93	96
DSJC500.1	5	12	16	18	22	18	22
DSJC500.5	13	48	67	70	80	70	77
DSJC500.9	59	126	161	172	181	172	177
DSJC1000.1	6	20	26	30	38	30	37
DSJC1000.5	14	83	114	123	143	122	136
DSJC1000.9	66	223	297	314	334	315	328
DSJR500.1	12	12	12	13	14	13	14
DSJR500.1c	80	85	87	102	111	102	109
DSJR500.5	119	122	130	142	156	141	150

Now we consider graphs reputed hard namely the Leighton, the random and the geometric graphs. The results are given in the tables IV and V where we compare the algorithms outcomes with the best known static coloring k^* and with the DSATUR results.

The MinG and RandG variants give less competitive results in comparison with the MaxG version. However, for all the instances the two methods FF and MaxG give almost the same results except for some instances namely le450-15d, DSJC1000.5 and DSJR500.5 where MaxG gives a better outcome then First-Fit.

Regarding the DSATUR results, we found that the results given by the proposed greedy algorithms are acceptable considering that the online algorithms have a partial view of the graph, since the vertices are given one by one.

VII. CONCLUSION

The dynamic graph coloring have many real life applications in which we deal with a partial or missing data at a given time t and yet have to take decisions (assign colors).

We proposed a generic greedy framework for online graph coloring. Different rules were tested on a large variety of graphs extracted from the ROIS benchmark.

We found that the proposed MaxG variant is efficient on almost all the treated instances, except for some categories of graphs. For these instances, the proposed online greedy algorithms still competitive with the static method DSATUR.

In the future work, we intend to investigate more rules for greedy algorithms to deal with these graphs reputed hard.

REFERENCES

[1] N. Barnier and P. Brisset, *Graph coloring for air traffic flow management*, Annals of Operations Research, 2004.

[2] A. Caprara, L. Kroon, M. Monaci, M. Peeters and P. Toth, *Passenger railway optimization*, Handbook in Operations Research and Management Science 14 Elsevier, Amsterdam, 2007.

[3] C. Lucet, F. Mendes and A. Moukrim, *An exact method for graph coloring*, Computers and Operations Research, 2006.
[4] P. Galinier and A. Hertz; *A survey on local search methods for graph coloring*, Computers and Operations Research, 2006.
[5] D. Brélaz, *New methods to color vertices of a graph*, Communications of the ACM 22, 1979.
[6] S. Vishwanathan, *Randomized online graph coloring*, Journal of Algorithms, 1992.
[7] A. Miller, *Online graph colouring*, Canadian Under graduate Mathematics Conference, 2004.
[8] R.M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations, New York, 1972.
[9] F. Harary and G. Gupta, *Dynamic graph models*, Mathematical and Computer Modeling, 1997.
[10] A. Ferreira, *On models and algorithms for dynamic communication networks: the case for evolving graphs*, in 4e rencontres francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL 2002), Mèze, 2002.
[11] E. Soedarmadji and R. McEliece, *A dynamic graph algorithm for the highly dynamic network problem*, In Proc. of the 1st IEEE Intl. Workshop on Foundation and Algorithms for Wireless Networking(FAWN), 2006.
[12] A. Dutot, F. Guinand, D. Olivier and Y. Pigné. *Graphstream: A tool for bridging the gap between complex systems and dynamic graphs*, In EPNACS:Emergent Properties in Natural and Artificial Complex Systems workshop, 2007.
[13] V. Boginski, S. Butenko and P.M. Pardalos, *Mining market data: a network approach*, Computers and Operations Research 33, 2006.
[14] M.J. Bommarito II, D.M. Katz, J.L. Zelner and J.H. Fowler, *Distance measures for dynamic citation networks*, Physica A: Statistical Mechanics and its Applications, 2010.
[15] K. Madduri and D.A. Bader, *Compact graph representations and parallel connectivity algorithms for massive dynamic network analysis*, International Parallel and Distributed Processing Symposium, 2009.
[16] M. Halldórsson and M. Szegedy, *Lower bounds for on-line graph coloring*, Proceedings of the third annual ACM-SIAM Symposium on Discrete Algorithms. Orlando Florida, 1992.
[17] M. Trick, *ROIS: Registry for Optimization Instances and Solutions*.[Online].Available: http://mat.tepper.cmu.edu/ROIS/