Analysis of a parallel MCMC algorithm for graph coloring with nearly uniform balancing

Donatello Conte, Giuliano Grossi, Raffaella Lanzarotti, Jianyi Lin, Alessandro Petrini

Analysis of a parallel MCMC algorithm for graph coloring with nearly uniform balancing

Donatello Conte[a], Giuliano Grossi[b,**], Raffaella Lanzarotti[b], Jianyi Lin[c], Alessandro Petrini[b]

[a] *Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT - EA 6300)*
64 Avenue Jean Portalis, 37000 Tours, France
[b] *Dipartimento di Informatica, Universitá degli Studi di Milano*
Via Celoria 18, I-20133 Milano, Italy
[c] *Department of Mathematics, Khalifa University of Science and Technology, Al Saada St.*
PO Box 127788, Abu Dhabi, United Arab Emirates

ABSTRACT

We propose the analysis of a scalable parallel MCMC algorithm for graph coloring aimed at balancing the color class sizes, provided that a suitable number of colors is made available. Firstly, it is shown that the Markov chain converges to the target distribution by repeatedly sampling from suitable proposed distributions over the neighboring colors of each node, independently and hence in parallel manner. We prove that the number of conflicts in the improper colorings genereted thoughout the iterations of the algorithm rapidly converges in probability to 0. As for the balancing, given to the complexity of the distributions involved, we propose a qualitative analysis about the balancing level achieved. Based on a collection of multinoulli distributions arising from the color occurrences within every node neighborhood, we provide some evidence about the character of the final color balancing, which results to be nearly uniform over the color classes. Some numerical simulations on big social graphs confirm the fast convergence and the balancing trend, which is validated through a statistical hypothesis test eventually.

1. Introduction

Graph theory is a branch of discrete mathematics which also has cross-disciplinary aspects with computer science as well as other branches of natural and physical sciences. It plays significant roles in modelling real-world problems and exhibits numerous applications in Pattern Recognition, Operation Research, Chemistry, Physics and Engineering disciplines among others.

Within this theory, the graph coloring problem is one of the oldest and among the most popular constraint satisfaction problems. Basically, it consists in finding an assignment of colors to the vertices of a given graph such that no two adjacent vertices share the same color. Graph coloring is extensively used in many Pattern Recognition applications: social networks problem such as Community Identification in Dynamic Social Networks (Tantipathananandh et al., 2007), summarization of social networks messages (Mosa et al., 2017), Improving Friends Matching in Social Networks (Murad et al., 2016), and for Collective Spammer Detection (Fakhraei et al., 2015). It can also be used for recording medical or biometric images (Lupaşcu et al., 2013; Cuculo et al., 2014) and for finding good resource allocation scheme for device-to-device (D2D) communications (Tsolkas et al., 2012; Comi et al., 2016) in modern wireless communication systems (Janis et al., 2009).

Moreover, recast balancing as graph coloring problem frequently appears in parallel scientific computing to identify equitable subsets of independent tasks that allow for a better utilization of hardware resources, essentially by overcoming inefficiency due to small color classes (Lu et al., 2017).

A common characteristic of these problems is that the graphs have very large size, thus requiring a speed up of the traditional greedy sequential coloring heuristics (Coleman and Moré, 1983) typically by introducing parallelization techniques. For instance, in (Deveci et al., 2016) and (Chen et al., 2017) the authors focus on thread scalability for hundreds or thousands of threads, showing that it is possible to achieve both good performance and high quality with massive parallelism. Thus, thread scalability in parallel algorithms, namely the ability of hard-

**Corresponding author
e-mail: grossi@di.unimi.it (Giuliano Grossi)

ware and software to deliver greater computational power when the amount of resources is increased, is crucial and widely exploited in this work.

In the literature, parallel graph coloring problem has been tackled by several approaches but, at the best of our knowledge, very few of them address the problem of balancing the color classes in parallel manner. One category of them is based on searching for a maximal independent set of vertices on a progressively shrunk graph and the concurrent coloring of the vertices in the found independent set. Often the independent set itself is computed in parallel using some variant of the Luby's algorithm (Luby, 1985). Examples of such approaches are Jones and Plassmann (1992); Gjertsen et al. (1996). Another category includes methods that color as many vertices as possible concurrently, tentatively tolerating potential conflicts, while detecting and solving conflicts afterwards (e.g. Boman et al. (2005)). Despite these solutions are effective in producing a proper coloring, generally minimizing the number of colors, they produce highly skewed color classes, undesirable for many applications, such as parallel job scheduling, that requires balancing among the classes. At the other extreme, one could search for a coloring being *equitable*, that is a coloring that guarantees that the sizes of any two color classes differ by at most one (Meyer (1973)). This constraint is computationally very demanding and somehow too stringent for practical applications; moreover class size constraints are known to usually raise computational hardness when performing partitioning (Bertoni et al., 2012). *Balanced* coloring relaxes the equitable constraint requiring that any two color class sizes differ at most by an integer l greater than 1. Few approaches have been proposed to tackle Balanced graph coloring (e.g. Robert et al. (1996); Lu et al. (2015)). However, the limit of these methods is still that they are intrinsically sequential thus not scalable, becoming unfeasible on large graph. A promising direction of research on graph coloring concerns the Markov Chain Monte Carlo (MCMC) methods that allow sampling from non analytic complex distributions. The idea is to define an ergodic Markov chain whose steady state distribution is defined over the set of colorings we wish to sample from. Within the framework of graph coloring using Markov chains several contributions have been proposed. In Jerrum (1995) a simple sequential solution based on the Glauber dynamics has been adopted. The Glauber dynamics produces a Markov chain on a proper coloring where at each step a random vertex v is recolored, choosing a color uniformly at random from the permissible ones.

To deal with large graphs, in Conte et al. (2019) we presented an algorithm based on MCMC method producing balanced graph coloring in a parallel way. Moreover, in that preliminary work we showed the effectiveness of this stochastic approach through experiments on random graphs. In this paper we provide some advances outlined in the following points:

- we analyze the convergence the proposed MCMC algorithm on the basis of the stochastic rules used in the different stages of the overall sampling process;
- we provide an intuitive and non-rigorous analysis of the balancing mechanism, whereas we assess the quality of the final balancing through a suitable asymptotic hypothesis test of statistical fitting;
- we have conducted experiments on social graphs (extracted from real data) in order to show that both convergence and balancing properties do not depend on "regular" topologies as those exhibited by random graphs.

As for experiments, to achieve significant speedups we leverage on modern many-core GPUs architectures (NVIDIA, 2019). Numerical results also show that the number of threads used by the parallel algorithm scales well with the graph sizes, even if compared with the GPU exploitation by the greedy strategy.

The remainder of the paper is organized as follows: Section 2 recall the basic principles of the MCMC algorithm described in Conte et al. (2019); the convergence analysis is given in Section 3 while in Section 4 we show numerical results on some real graphs of considerable size. Section 5 concludes the paper.

2. A parallel MCMC algorithm for graph coloring

In this section we briefly outline the parallel algorithm for graph coloring based on the MCMC sampling technique presented in Conte et al. (2019).

2.1. Notations

We will consider a simple undirected graph $G = \langle V, E\rangle$ with $n = |V|$ vertices and the set $[k] = \{1, \dots, k\}$ of colors used to label the vertices. A k-coloring $c : V \to [k]$, also represented as vector $c = (c_1, \dots, c_n) \in [k]^n$, is called *proper* if adjacent vertices receive different colors, otherwise it is termed *improper*. It is well-known that, if $\Delta(G)$ is the maximum degree of G, $k = \Delta(G) + 1$ colors are sufficient to properly color the graph by a sequential greedy algorithm. For a given coloring c, let $\mathcal{N}(v)$ denote the neighborhood of node v in G, and $c_{\mathcal{N}(v)} \subseteq [k]$ be the set of colors occupied by vertices $\mathcal{N}(v)$ and $\bar{c}_{\mathcal{N}(v)}$ its complement; let us denote the respective cardinalities with $h_v(c) = |c_{\mathcal{N}(v)}|$ and $\bar{h}_v(c) = |\bar{c}_{\mathcal{N}(v)}|$. Given c, an edge $uv \in E$ with $c_u = c_v$ is a *conflict* and $\#(c) : [k]^n \to \mathbb{N}$ counts the number of conflicts. We will also consider the absolute *frequency* of the color j in c: $f_j(c) = |\{u \in V : c_u = j\}|$.

Hereafter we will use lowercase letters, e.g. c, c', c^*, for given colorings and uppercase for random colorings, e.g. C, C', C^*. For example the probability of $C' = c'$ given $C = c$ will be denoted $\mathbb{P}(C' = c' \mid C = c)$ or $\mathbb{P}(c' \mid c)$ for short.

2.2. Markov Chain Monte Carlo for sampling colorings

This novel parallel Markov Chain Monte Carlo (MCMC) technique relies on a 1st-order ergodic Markov chain $(C^i)_{i=1}^{\infty}$ visiting a sequence of (possibly improper) k-colorings $c \in [k]^n$ and whose stationary distribution π strongly depends on the set of conflicts involved in c. As usual, we consider the *Gibbs distribution* as target stationary distribution for the Markov chain:

$$\pi(c) = \frac{e^{-\beta \#(c)}}{Z(\beta)}, \quad \text{with } Z(\beta) = \sum_{c' \in [k]^n} e^{-\beta \#(c')}. \tag{1}$$

The role of parameter β in (1) aims to penalize improper colorings, leading π toward the uniform distribution over the proper colorings only with exponential decrease (as β increases). It is known from MCMC theory that the latter distribution is asymptotically approached in the sampling process when the chain is suitably constructed using the well-established Metropolis-Hastings algorithm (Hastings, 1970). This construction requires the specification of a *proposal* probability encapsulating the *acceptance ratio* and a *transition* probability for the chain.

As for the transition probabilities of the Markov chain, given a coloring $C = c$ we sample the successive coloring C^* in two phases acting according to a typical "rejection sampling" scheme (Voss, 2013): first a candidate coloring C' is generated according to a suitable proposal probability $r(c, c') := \mathbb{P}(c' \mid c)$, then the proposal C' is accepted effectively as successive coloring C^* according to the acceptance ratio $\alpha(c, c') := \min\left\{\frac{\pi(c')r(c',c)}{\pi(c)r(c,c')}, 1\right\}$:

$$\mathbb{P}(c' \mid c) := \begin{cases} \alpha(c, c'), & c' \neq c \\ 1 - \alpha(c, c'), & \text{otherwise} \end{cases}.$$

The proposal coloring C' is sampled with probability $r(c, c')$ as follows. Each node $v \in V$ is drawn independently and with identical distribution $\mathbb{P}(c'_v \mid c)$ of colors so that the overall proposal probability is

$$r(c, c') = \prod_{v \in V} \mathbb{P}(c'_v \mid c). \tag{2}$$

Notice that, in the construction of the acceptance ratio also the *backward* probability $r(c', c)$ is required, hence $r(c, c')$ is called *forward* probability.

The choice of the node proposal probability $\mathbb{P}(c'_v \mid c)$ is a key step and is hence detailed distinctly in the following subsection. It is also important to observe from the computational viewpoint that the independent drawing of all c'_v, $v \in V$, allows for the generation of the new coloring in a parallel manner.

2.3. Proposal distribution for new colorings

Here we specify the algorithm for the proposal distribution procedure so that the stochastic evaluations follow consequently from the analysis of the color generation. First, the behavior of the algorithm splits into two cases based on the old coloring c. When there is some conflict locally for v, namely $c_v \in c_{\mathcal{N}(v)}$, the new proposed color C'_v for v shall be redrawn with the aim of reducing the possible conflicts. We draw it from the *free* colors $\bar{c}_{\mathcal{N}(v)}$ following a nearly uniform distribution of $C'_v = j$ given c:

$$\eta_v(j, c) = \begin{cases} \frac{1-\varepsilon}{k-h_v(c)}, & \text{if } j \in \bar{c}_{\mathcal{N}(v)} \\ \varepsilon, & \text{if } j \in c_{\mathcal{N}(v)}. \end{cases} \tag{3}$$

The rationale behind such definition is that we want to generate with high probability, a color equally likely among those free, in order to aim at the balancing objective of the method. Nevertheless, we keep a negligible chance $\varepsilon > 0$ to pick a color that is not free, in order to widen the search space.

As for the case of no conflict for v, i.e. $c_v \in \bar{c}_{\mathcal{N}(v)}$, it is desirable to keep nearly surely the old color c_v to facilitate the convergence of the algorithm, or otherwise pick another color with a small chance ε. Hence, in the case of no conflict, for the node v the proposal color $C'_v = j$ given c is distributed as

$$\zeta_v(j, c) = \begin{cases} 1 - \varepsilon\,(k-1), & \text{if } j = c_v \\ \varepsilon, & \text{if } j \neq c_v. \end{cases} \tag{4}$$

With the above definitions we derive the following conditional distribution for proposal color (also called forward):

$$\mathbb{P}(c'_v \mid c) = \begin{cases} \eta_v(c'_v, c), & \text{if } h_v(c) < k, c_v \in c_{\mathcal{N}(v)} \\ \zeta_v(c'_v, c), & \text{otherwise.} \end{cases} \tag{5}$$

The backward probabilities

$$r(c', c) = \mathbb{P}(C' = c \mid C = c') = \prod_{v \in V} \mathbb{P}(C'_v = c_v \mid C = c')$$

can be obtained by symmetrical reasoning, i.e. exchanging the role of c and c' in the calculations outlined above. This allows to compute then the acceptance ratio $\alpha(c, c')$.

The main procedural steps of the MCMC algorithm described above are sketched in Algorithm 1.

Algorithm 1: Parallel MCMC Graph Coloring

Input : Graph $G = \langle V, E \rangle$ with $n = |V|$;
Number k of colors;
Gibbs parameter $\beta \ll 1$
Output: Random proper coloring $C \in [k]^n$

$C \leftarrow$ random initial coloring $\in [k]^n$
while $\#(C) > 0$ **do**
  foreach $v \in V$ **in parallel do**
    Calculate $C_{\mathcal{N}(v)}$
    $h_v(C) \leftarrow |C_{\mathcal{N}(v)}|$
    $\mathbb{P}(c'_v \mid C) \leftarrow$ Compute according to (5)
    $C'_v \leftarrow$ Generate with distribution $\mathbb{P}(c'_v \mid C)$
  $C' \leftarrow$ Proposed coloring $(C'_1, C'_2, ..., C_n)$
  $r(C, C') \leftarrow$ Compute forward probability
  $r(C', C) \leftarrow$ Compute backward probability
  $\alpha(C, C') \leftarrow \min\left\{\frac{r(C',C)}{r(C,C')} e^{-\beta(\#(C')-\#(C))}, 1\right\}$
  Accept $C \leftarrow C'$ with probability $\alpha(C, C')$

3. Algorithm analysis

We now deal with the study of the behavior of the highly scalable parallel MCMC algorithm for graph coloring proposed in Conte et al. (2019), and outlined in the previous section. First, we claim that in the Markov chain, the color distribution stochastically converges to proper colorings by repeatedly sampling a pool of easier proposal distributions built upon the sets of neighboring node colors. Then, we also develop a discussion about the quality of balancing achieved, the latter being the main goal of this modelling.

3.1. Convergence analysis

As for the convergence properties of Algorithm 1, due to the intrinsic randomness in drawing colors we cannot guarantee that the number of conflicts $\#C$ strictly decreases at each iteration. Therefore we give a characterization of the convergence in probabilistic terms, studying a slight variant of Algorithm 1 where we bring the parameters to the extreme values: acceptance ratio $\alpha(C, C') = 1$ and $\varepsilon = 0$.

Lemma 1. *Given current coloring C, let $\#W$ and $\#W^*$ be number of nodes in G having some conflict at, respectively, any current and successive iteration of Algorithm 1, where we set acceptance ratio $\alpha(C, C') = 1$ and $\varepsilon = 0$. For any integer $r > 1$ and provided a number of colors $k \geq \Delta(G) + r$, the following expectation inequality holds:*

$$\mathbb{E}[\#W^* \mid C] \leq \rho \#W$$

for some $0 < \rho = 1 - (1 - 1/r)^{\Delta(G)} < 1$.

Proof. Preliminarily, we introduce the convenient notation of *indicator* expression $\mathbb{1}(cond)$ that takes value 1 if the condition *cond* is true, and 0 otherwise. The number of conflicts in the new coloring C^* can be written as $\#C^* = \sum_{uv \in E} \mathbb{1}(C^*_u = C^*_v)$, while the number of *local* conflicts is $\#_v C^* = \sum_{u:uv \in E} \mathbb{1}(C^*_u = C^*_v)$; similar definitions can be given for $\#C$ and $\#C_v$. Given the current coloring C we partition V into nodes with conflicts, $W = \{u \in V : \#_u C > 0\}$, and conflict-free nodes $\bar{W} = V \setminus W$.

First, notice that the number of local conficts at any $v \in W$ is simply

$$\#_v C^* = \sum_{u \in W : uv \in E} \mathbb{1}(C^*_u = C^*_v)$$

due to $\sum_{u \in \bar{W} : uv \in E} \mathbb{1}(C^*_u = C^*_v) = 0$, since when u is not in W it maintains the old color $C^*_u = C_u \neq C^*_v$. Hence, in the summation in $\#_v C^*$ each term $\mathbb{1}(C^*_u = C^*_v)$ is a Bernoullian variable with parameter

$$p_{uv} := \frac{|\bar{C}_{\mathcal{N}(u)} \cap \bar{C}_{\mathcal{N}(v)}|}{|\bar{C}_{\mathcal{N}(u)}| \cdot |\bar{C}_{\mathcal{N}(v)}|} \leq \frac{\min\left\{|\bar{C}_{\mathcal{N}(u)}|, |\bar{C}_{\mathcal{N}(v)}|\right\}}{|\bar{C}_{\mathcal{N}(u)}| \cdot |\bar{C}_{\mathcal{N}(v)}|} \leq \frac{1}{r},$$

conditioned on C, i.e.

$$\mathbb{1}(C^*_u = C^*_v) \mid C \sim \text{Bernoulli}(p_{uv}).$$

Moreover, these terms are stochastically independent since the drawing of C^*_u for each neighbor u of v is done autonomously from the other neighbors. Given C, the variable $\#_v C^*$ is hence a sum of independent Bernoullian variables having parameters p_{uv} with $uv \in E, u \in W$, namely a so-called Poisson-Binomial random variable:

$$\#_v C^* \mid C \sim \text{PoissonBinomial}(\{p_{uv} : uv \in E, u \in W\})$$

We define the Bernoulli variable $B^*_v := \mathbb{1}(\#_v C^* > 0)$ indicating whether v has some conflict in the new coloring. Thanks to the observation above, its distribution conditioned on C is easily determined as

$$(B^*_v \mid C) = (\mathbb{1}(\#_v C^* > 0) \mid C) \sim \text{Bernoulli}(1 - \Pi_{u \in W : uv \in E}(1 - p_{uv}))$$

when $v \in W$, since $\Pi_{u \in W : uv \in E}(1 - p_{uv})$ is the probability that v does not have any conflict with the neighbors in the new coloring C^*. Clearly, $B^*_v = 0$ if $v \notin W$. The number $\#W^* = \#\{u \in V : \#_u C^* > 0\} = \sum_{v \in W} B^*_v$ of conflicting nodes in the new coloring C^* thus satisfies the following relationships

$$\begin{aligned}
\mathbb{E}[\#W^* \mid C] &= \mathbb{E}\left[\sum_{v \in W} B^*_v \mid C\right] = \sum_{v \in W} \mathbb{E}[B^*_v \mid C] \\
&= \sum_{v \in W} \left[1 - \prod_{u \in W : uv \in E} (1 - p_{uv})\right] \\
&\leq \sum_{v \in W} \left[1 - \prod_{u \in W : uv \in E} \left(1 - \frac{1}{r}\right)\right] \\
&\leq \sum_{v \in W} \left[1 - \left(1 - \frac{1}{r}\right)^{\deg v}\right] \leq \left[1 - \left(1 - \frac{1}{r}\right)^{\Delta(G)}\right] \sum_{v \in W} B_v \\
&= \left[1 - \left(1 - \frac{1}{r}\right)^{\Delta(G)}\right] \#W = \rho \#W
\end{aligned}$$

where $0 < \rho := 1 - (1 - 1/r)^{\Delta(G)} < 1$. That is, the claim $\mathbb{E}[\#W^* \mid C] \leq \rho \#W$ holds. □

Directly following the previous result, we have the following convergence.

Theorem 1. *The number of conflicts $\#C$ converges in probability to 0, i.e. $\lim_{t \to \infty} \mathbb{P}(\#(C^t) < \delta) = 1 \ \forall \delta > 0$.*

Proof. From Lemma 1, using the monotonicity property of the expected value we have

$$\mathbb{E}[\mathbb{E}[\#W^* \mid C]] \leq \rho \mathbb{E}[\#W].$$

By the Tower Rule of the conditional expectation ($\mathbb{E}[\mathbb{E}[\#W^* \mid C]] = \mathbb{E}[\#W^*]$) we can give the bound

$$\mathbb{E}[\#W^*] \leq \rho \mathbb{E}[\#W].$$

The coloring C^* is the one successive to C; hence building the sequence of successive colorings C^t, $t = 0, 1, 2, ...$ by means of the algorithm above, one can guarantee $\mathbb{E}[\#W^t] \leq \rho^t \mathbb{E}[\#W^0]$ for any t. Hence, applying $\lim_{t \to \infty}$ to both side, we have the convergence in expectation of the number of conflicting nodes to 0:

$$\lim_{t \to \infty} \mathbb{E}[\#W^t] = \lim_{t \to \infty} \rho^t \mathbb{E}[\#W^0] = 0.$$

Now, it is well known that convergence in expectation (namely, L^1-convergence in the proper probability space $(\Omega, \mathcal{F}, \mathbb{P})$) implies the convergence in probability, i.e.

$$\lim_{t \to \infty} \mathbb{P}(\#W^t < \delta) = 1 \qquad \text{for any } \delta > 0.$$

Since the number $\#C^t$ of conflicts is easily bounded above by $\frac{1}{2}\Delta(G)\#W^t$, it converges in probability to 0 as well:

$$\lim_{t \to \infty} \mathbb{P}(\#C^t < \delta) = 1 \qquad \text{for any } \delta > 0.$$

□

3.2. Qualitative analysis of the balancing

The balancing of color classes is an important aspect but it is difficult to prove it rigorously, due to the complex role that the distributions arising from the model play in the evolution of the algorithm towards attaining the final coloring. Nevertheless, in this section we sketch a qualitative non-rigorous analysis of this process in order to provide some intuition and rationale about the nearly uniform character of final color frequencies, regardless of the topology of the graphs at hand.

To carry out this analysis, we resort to the (n,k)-Poisson Multinomial Distribution (PMD) (Daskalakis et al., 2015) which is the distribution of the sum of n independent random variables supported on the set $\mathcal{B}_k = \{e_1, \ldots, e_k\}$ of standard basis vectors in $\mathbb{R}^k$, representing the k colors provided to the algorithm. We replace the representation of the color set $[k]$ with $\mathcal{B}_k$ for technical reasons in this section only.

At each iteration step t of the Algorithm 1, each node $v \in V$ randomly draws a color with probability distribution (5). Let $F_v^{(t)} \subseteq \mathcal{B}_k$ be the set of free colors, at step t, for node v whose color is represented by the random variable $C_v^{(t)} \in \mathcal{B}_k$, and let $p_v^{(t)} = \left(p_{v,1}^{(t)}, \ldots, p_{v,k}^{(t)}\right)$ be the distribution of the new color for v in $\mathcal{B}_k$. Clearly, due to the random drawing triggered by (5) we have

$$p_{v,j}^{(t)} \approx \begin{cases} 1/|F_v^{(t)}|, & \text{if } e_j \in F_v^{(t)} \\ 0, & \text{if } e_j \notin F_v^{(t)} \end{cases} \tag{6}$$

in case of conflict for v (eq. (3)), or

$$p_{v,j}^{(t)} \approx \begin{cases} 1, & \text{if } e_j = C_v^{(t)} \\ 0, & \text{otherwise} \end{cases}, \tag{7}$$

when v is conflict-free (eq. (4)). This means that the process of assigning a new color $C_v^{(t+1)} \in \mathcal{B}_k$ to node v, can be expressed by a multinoulli distribution with parameter $p_v^{(t)}$, i.e.,

$$C_v^{(t+1)} \sim \text{Multinoulli}(p_v^{(t)}).$$

Notice that each node in the color $C_v^{(t+1)}$ is drawn independently from each other.

Let $X^{(t)} = \sum_{v \in V} C_v^{(t)}$ be the number of occurrences of all colors in the graph coloring at time t. In particular, for each $j \in [k]$, $X_j^{(t)}$ represents the frequency of color e_j within the coloring $C^{(t)}$. Hence, the random vector $X^{(t)}$ is the sum of independent but not identically distributed multinoulli random variables, which follows a (n,k)-PMD, where n is the number of vertices and k the number of colors. Let $\xi(x) = \mathbb{P}(X^{(t)} = x)$ denote the probability mass function of $X^{(t)}$, where x is a k-dimensional vector with non-negative integer entries summing up to n. Clearly, in order to have the desired balancing, it is crucial to assign high probability $\xi(x)$ to the elements x with equalized entries. In the following, we provide a qualitative and non-rigorous analysis of the dynamics of color assignments and its distribution evolution in order to support the experimental evidence that the color classes are quite balanced among each other. Let us investigate the following steps in the coloring process generated by Algorithm 1.

Step 1. Assume that in the first step the colors are assigned independently and uniformly at random at each node. As a consequence, the set of free colors $F_v^{(1)}$ of v comes up uniformly among all subsets of size $\left|F_v^{(1)}\right|$. Let us observe how a new coloring is generated by focusing on a color e_j through all the n independent trials, each one corresponding to a node v. To consider the overall chances e_j has, we have to focus in turn on the probability sets $P_j = \left\{p_{v,j}^{(1)} : v \in V\right\}$. In particular, for large n and under the above independence assumptions, it results that $p_{v,j}^{(1)}$ is the same on average for all v due to the uniformness of $F_v^{(1)}$, so that e_j is assigned nearly the same chance to be selected in each trial. In other words, the averages over all nodes $\bar{P}_j$ should be approximately $1/k$ for large graph size n.

Step t. Assuming that at time $t-1$ the (in general improper) coloring $C^{(t-1)}$ is quite balanced in terms of color frequencies, we can conclude that also the new coloring $C^{(t)}$ could achieve a good balancing. This should happen thanks to the following two facts. First, some colors are definitely fixed since they are randomly chosen in some previous step, as described by the (7) and this contribute to uniformly spread the colors in the neighborhoods of those nodes that have to make a choice. Second, those nodes that have to update the color, are in the same condition described in Step 1, thus they repeat the same independent drawings.

Figure 1 provides an empirical demonstration of the balancing trend exhibited by the MCMC algorithm and described in the above steps for a social graph (labeled *ca-coauthors-dblp*) introduced in the next section on numerical simulations. In fact, the balancing quality in each repetition (light blue lines) and on average (red line) is maintained quite constant over all iterations (**Step t**) apart from the first iteration (**Step 1**) that requires an adjustment as a consequence of the random initial coloring.

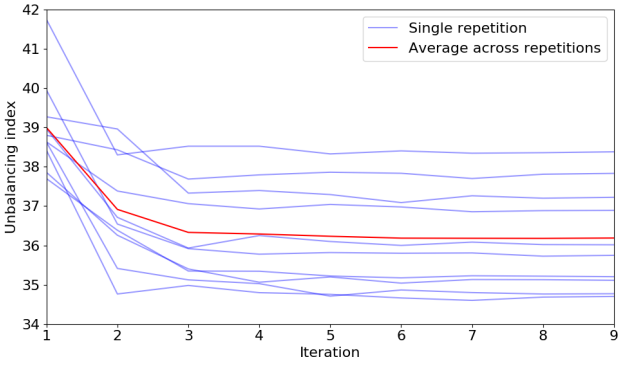

Figure 1: Color balancing quality of a sample graph (the social graph *ca-coauthors-dblp*) during the iterations of the parallel MCMC algorithm for the first 10 states (colorings) visited by the Markov chain.

4. Numerical simulations

In Conte et al. (2019) we offered a preliminary assessment of a novel parallel stochastic technique for recovering balanced

colorings on big random graphs. Here we extend the evaluation to a small sample of real-world instances, where graphs describe relations among individuals, to identify differences and analogies in behavior between synthetic and real-world data. A second purpose of these experiments is to empirically validate both algorithm convergence and balancing properties in the light of the analysis delineated in the previous section.

Hence, as in Conte et al. (2019), we tested our GPU implementation of the presented strategy, hereafter called MCMC-GPU, against a non-parallel implementation (called MCMC-CPU) of the same algorithm which runs on a standard CPU. The purpose of this comparison is to evaluate the scalability and speed-up of the parallel approach taking the sequential implementation as baseline reference. Also, we tested the MCMC-GPU implementation against a fast and fully parallel greedy coloring strategy inspired by Luby (1985) work, called Luby-GPU. This simple coloring strategy provides a proper coloring in very short time, at the cost of a very high class unbalance. All the algorithms were implemented[1] in C++ and both MCMC-GPU and Luby-GPU leverages the NVidia CUDA programming paradigm (NVIDIA (2019)). All the host code is single-threaded and it is meant to be run on a single machine, hence no other libraries (such as pthread, C++11 threads, OpenMP, MPI, etc...) are required. Both CPU and GPU code is compiled with NVidia nvcc v10.1, using the underlying GNU GCC v5.4.0 as C++ compiler.

For running the experiment we used a workstation with 2× Intel Xeon CPU E5-2620 v3 @ 2.40GHz, 64GB of RAM, Linux 16.04 and an NVidia GTX980 GPU featuring 2880 cores and 4GB of VRAM. MCMC-CPU runs are single-threaded, hence were run on a single core of the host CPU. For Luby-GPU and MCMC-GPU, on the host the processes are single-threaded as well, and parallelization occurs only on the GPU side with the 2880 cores.

The three coloring approaches have been tested on four real-world social graphs taken from the Rossi and Ahmed (2016) repository. They greatly vary in terms of size (number of nodes and edges) and average/maximum degree, thus providing a small but relatively diverse array of use cases. Their relevant features are summarized in Table 1. The maximum number of possible colors to be used is a parameter in our algorithm and it has to be fixed in advance. We fixed this parameter in such a way that the results are stable for each trial. Table 1 shows the values of this parameter for each graph used in the experiment.

For measuring the overall balancing quality of a coloring c, we used the following

quantity, we call

unbalancing index, which is closely related to the

classical root mean square deviation

of the color class sizes $\{n_j : j = 1, \ldots, k\}$:

$$\Gamma_{n,k}(c) = \left(\frac{1}{k} \sum_{j=1}^{k} \left| n_j - \frac{n}{k} \right|^2 \right)^{1/2} . \qquad (8)$$

[1]The software used for the experiments is freely available on the GitHub repository: `https://github.com/phuselab/MCMC_Colorer`

Naturally, a perfectly balanced coloring satisfies $\Gamma_{n,k}(c) = 0$.

Figure 2 top shows the average unbalancing index over 35 repetitions for each coloring algorithm applied to the 4 sample graphs. Giving to its greedy design, Luby-GPU achieved a worse balancing performance, being largely outperformed by MCMC strategy. For the latter we set a number of colors a bit greater than the one obtained by Luby (for details see Table 1).

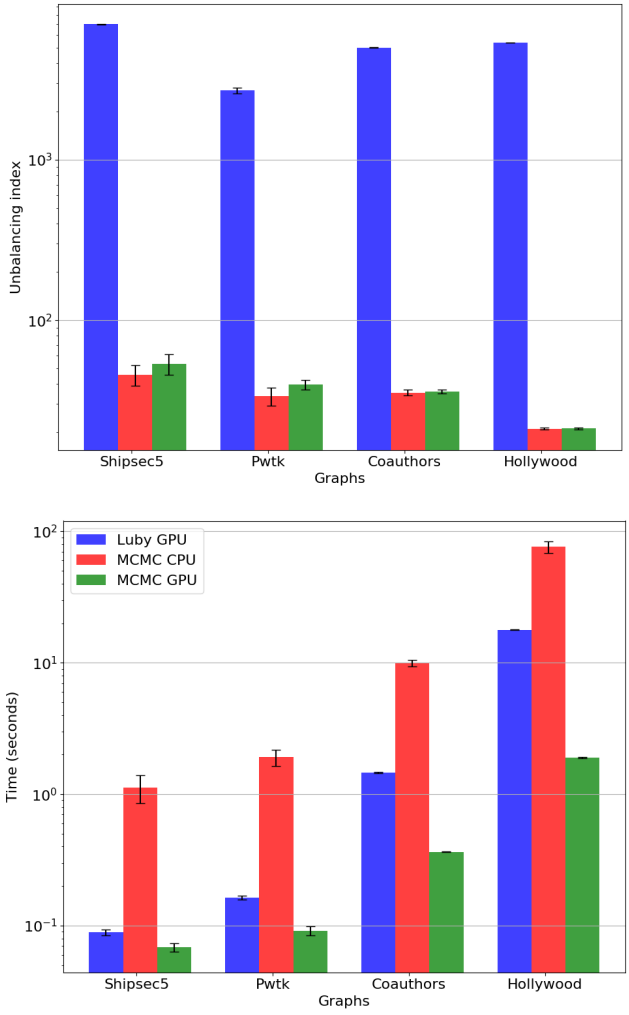

Figure 2: Average unbalancing index (upper plot) of final colorings achieved by the three algorithms for the graphs of Table 1 and related average execution times (lower plot), expressed in seconds. Results are on 35 repetitions (standard deviation as error bar) and plotted in logarithmic scale.

Figure 2 bottom shows the average computing times over the trials. Being inherently sequential, MCMC-CPU is the slowest and a direct comparison with MCMC-GPU results in an average speed-up ranging from 28× (for *sc-pwtk*) up to 69× (for *ca-hollywood-2009*). We also remark that in our experiments, MCMC-GPU produces a proper coloring for each graph in less time than Luby-GPU does, showing that sometimes it is even faster than the greedy approach.

For statistically assessing the balancing, we test the uniformness of color distribution through a hypothesis test starting

Table 1: Features of the graphs used in the experiments: number of nodes and edges (columns 2-3), maximum and average degree (columns 4-5), average number of colors found by Luby-GPU (column 6) and the parameter of maximum number of colors to be used, set for MCMC-GPU (and MCMC-CPU) (last column).

Graph	Nodes	Edges	Max deg	Avg deg	N. colors by Luby	N. colors by MCMC-GPU
sc-shipsec5	179.1K	4.4M	75	24	33	50
sc-pwtk	218K	11.4M	179	51	43	85
ca-coauthors-dblp	540K	30M	3.3K	56	337	460
ca-hollywood-2009	1.1M	56M	11.5K	105	2209	2400

from the balancing index (8). Given a coloring c produced by MCMC-GPU, we consider the statistic defined as

$$K^2 = \frac{(k\,\Gamma_{n,k}(c))^2}{n} = \sum_{j=1}^{k} \frac{(n_j - n/k)^2}{n/k}.$$

In fact, this expression is well known in statistical inference as the Pearson's chi-squared statistic for goodness-of-fit test (Mood et al., 1973, §IX.5.2), that is used for assessing whether the observed categorical outcomes of the experiment have frequencies (n_j) following a hypothesized categorical distribution, namely uniform distribution in our case (i.e. color class sizes n/k). With this so-called null hypothesis H_0, it was proved using the Central Limit Theorem and a suitable geometric transformation (see Cochran (1952)) that such statistic K^2 has asymptotically a $\chi^2(k-1)$ distribution (chi-squared distribution with $k-1$ degrees of freedom) as $n \to \infty$: hence, $\chi^2(k-1)$ is a good approximation of the true distribution for large number n of nodes, as in our experiments.

The key point is that the statistic K^2 is large and the corresponding p-value is small, when the produced color class sizes are unbalanced, and vice-versa.

Results of the test are reported in Table 2 for 35 runs: notice that the range of rather large p-values leads to accept the H_0 hypothesis stating the color balancing in all analyzed graphs.

Table 2: Test for fitting uniform color distribution on the result of 35 runs with the analyzed graphs. Ranges of observed Pearson's chi-squared test-statistics and corresponding p-values are shown.

Graph	K^2 statistic range	p-value range
sc-shipsec5	[19.51, 50.52]	[0.413, 0.999]
sc-pwtk	[31.14, 64.57]	[0.940, 1.000]
ca-coauthors-dblp	[368.09, 473.42]	[0.310, 0.998]
ca-hollywood-2009	[2205.43, 2446.97]	[0.242, 0.997]

5. Conclusions and discussion

We have studied the convergence properties of a scalable parallel algorithm for graph coloring problem based on Markov Chain Monte Carlo techniques (Conte et al., 2019). We also have sketched a qualitative analysis of the achieved balancing, remarking that it is as important as difficult to formally derive an analytic form for the distribution on the final colorings, due to the complex dynamics among the applied probability distributions that the model brings into play during its evolution. It would be also worth it to study MCMC models that incorporate in the target distribution not only the term promoting proper colorings (as in the current one), but also new specific terms helping in recovering suitable families of colorings not necessarily equalized in the color classes. For example, in case of application of graph coloring to the deployment of independent tasks performed concurrently on a pool of processors (typically within datacenters or on distributed architectures), in order to gain computational efficiency the target distribution should look for colorings having color classes of size multiple of the number of processors.

In Conte et al. (2019) the effectiveness of this stochastic approach in finding balanced colorings has been shown through experiments on random graphs. Here we have extended the algorithm assessment on some kind of social graphs just to show that both convergence and balancing properties are preserved although the topologies of the graphs considered are not as "regular" as those exhibited by random graphs. As for the experimental analysis, further studies on realistic applications where graph topology and size (especially big graphs) are taken into account should be conducted. For instance, it is important to stress the heuristic ability on graphs where edge density is not uniform and the presence of quasi-clique subsets interferes with the needs of MCMC model to locally find low conflict level with high probability.

Moreover, being an heuristic technique based on a number of parameters that affect the behavior and ultimately its ability to find highly balanced solutions, a both theoretical and experimental deep analysis focused on some of the most relevant parameters is worth a great attention. For instance, it is the case of the number of colors required as input to the MCMC algorithm, which gives rise to a clear trade off between convergence speed and balancing quality, as emerged since the early experiments. There are also meta-parameters related to the MCMC technique itself that affect the results, such as the parameter β appearing in the formulation of the target stationary distribution for the Markov chain (Eq. 1). Furthermore, besides its effect in penalizing improper colorings and leading π toward the uniform distribution over the proper ones, β or some function of it would also have a relevant role in defining new interesting metrics that can promote colorings that capture graph properties other than the balancing, such as those associated to planar graphs.

References

Bertoni, A., Goldwurm, M., Lin, J., Saccà, F., 2012. Size constrained distance clustering: separation properties and some complexity results. Fundamenta Informaticae 115, 125–139.

Boman, E.G., Bozdağ, D., Catalyurek, U., Gebremedhin, A.H., Manne, F., 2005. A scalable parallel graph coloring algorithm for distributed memory computers, in: Euro-Par 2005 Parallel Processing, pp. 241–251.

Chen, X., Li, P., Fang, J., Tang, T., Wang, Z., Yang, C., 2017. Efficient and high-quality sparse graph coloring on gpus. Concurrency and Computation: Practice and Experience 29, e4064.

Cochran, W.G., 1952. The $\chi 2$ test of goodness of fit. The Annals of mathematical statistics , 315–345.

Coleman, T.F., Moré, J.J., 1983. Estimation of sparse Jacobian matrices and graph coloring problems. SIAM Journal on Numerical Analysis 20, 187–209.

Comi, P., Crosta, P.S., Beccari, M., Paglierani, P., Grossi, G., Pedersini, F., Petrini, A., 2016. Hardware-accelerated high-resolution video coding in virtual network functions, in: Eu. Conf. on Network and Communication, pp. 32–36.

Conte, D., Grossi, G., Lanzarotti, R., Lin, J., Petrini, A., 2019. A Parallel MCMC Algorithm for the Balanced Graph Coloring Problem, in: Conte, D., Ramel, J.Y., Foggia, P. (Eds.), Graph-Based Representations in Pattern Recognition, Springer. pp. 161–171.

Cuculo, V., Lanzarotti, R., Boccignone, G., 2014. Using sparse coding for landmark localization in facial expressions, in: 5th European Workshop on Visual Information Processing, EUVIP 2014, IEEE. pp. 1–6.

Daskalakis, C., Kamath, G., Tzamos, C., 2015. On the structure, covering, and learning of poisson multinomial distributions, in: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, pp. 1203–1217.

Deveci, M., Boman, E.G., Devine, K.D., Rajamanickam, S., 2016. Parallel graph coloring for manycore architectures, in: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 892–901. doi:10.1109/IPDPS.2016.54.

Fakhraei, S., Foulds, J., Shashanka, M., Getoor, L., 2015. Collective spammer detection in evolving multi-relational social networks, in: Proc. of the 21th Int. Conf. ACM SIGKDD, pp. 1769–1778.

Gjertsen, R.K., Jones, M.T., Plassmann, P.E., 1996. Parallel heuristics for improved, balanced graph colorings. J. Par. and Dist. Comput 37, 171–186.

Hastings, W., 1970. Monte carlo sampling methods using markov chains and their applications. Biometrika 57, 97–109.

Janis, P., Chia-Hao, Y., Doppler, K., Ribeiro, C., Wijting, C., Klaus, H., Tirkkonen, O., Koivunen, V., 2009. Device-to-device communication underlaying cellular communications systems. Int. Jour. of Comm., Network and System Sciences 2-3.

Jerrum, M.R., 1995. A very simple algorithm for estimating the number of k-colourings of a low-degree graph. Random Structures and Algorithms 7, 157–165.

Jones, M.T., Plassmann, P.E., 1992. A parallel graph coloring heuristic. SIAM J. Sci. Comput. 14, 654–669.

Lu, H., Halappanavar, M., Chavarria-Miranda, D., Gebremedhin, A., Kalyanaraman, A., 2015. Balanced coloring for parallel computing applications, in: 29th IEEE Int. Parallel and Distributed Processing Symposium, pp. 7–16.

Lu, H., Halappanavar, M., Chavarria-Miranda, D., Gebremedhin, A.H., Panyala, A., Kalyanaraman, A., 2017. Algorithms for balanced graph colorings with applications in parallel computing. IEEE Transactions on Parallel and Distributed Systems 28, 1240–1256. doi:10.1109/TPDS.2016.2620142.

Luby, M., 1985. A simple parallel algorithm for the maximal independent set problem, in: Proc. 17th Annual ACM Symposium on Theory of Computing, pp. 1–10.

Lupaşcu, C.A., Tegolo, D., Bellavia, F., Valenti, C., 2013. Semi-automatic registration of retinal images based on line matching approach, in: Proc. of the 26th IEEE Int'l Symposium on Computer-Based Medical Systems, IEEE. pp. 453–456.

Meyer, W., 1973. Equitable coloring. Amer. Math. Monthly 80, 920–922.

Mood, A.M.., Graybill, F.A., Boes, D.C., 1973. Introduction to the Theory of Statistics. McGraw-Hill.

Mosa, M.A., Hamouda, A., Marei, M., 2017. Graph coloring and aco based summarization for social networks. Expert Systems with Applications 74, 115–126.

Murad, O., Sleit, A., Sharaiah, A., 2016. Improving friends matching in social networks using graph coloring. International Journal 15.

NVIDIA, 2019. NVIDIA CUDA Toolkit 10.0. URL: https://developer.nvidia.com/cuda-toolkit.

Robert, J., Gjertsen, K., Jones, M.T., Plassmann, P., 1996. Parallel heuristics for improved, balanced graph colorings. Journal of Parallel and Distributed Computing 37, 171–186.

Rossi, R.A., Ahmed, N.K., 2016. An interactive data repository with visual analytics. SIGKDD Explor. 17, 37–41.

Tantipathananandh, C., Berger-Wolf, T., Kempe, D., 2007. A framework for community identification in dynamic social networks, in: Proc. 13th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining, ACM. pp. 717–726.

Tsolkas, D., Liotou, E., Passas, N., Merakos, L., 2012. A graph-coloring secondary resource allocation for d2d communications in lte networks, in: IEEE 17th Int. Workshop on Computer Aided Modeling and Design (CAMAD), IEEE. pp. 56–60.

Voss, J., 2013. An introduction to statistical computing: a simulation-based approach. John Wiley & Sons.