A parallel MCMC algorithm for the Balanced Graph Coloring problem

D. Conte[1], G. Grossi[2], R. Lanzarotti[2], J. Lin[3], and A. Petrini[2]

[1] Université de Tours, Computer Science Laboratory (LIFAT - EA6300)
64 Avenue Jean Portalis, 37000 Tours, France
donatello.conte@univ-tours.fr
[2] Dipartimento di Informatica, Università degli Studi di Milano
Via Celoria 18, I-20133 Milano, Italy
grossi@di.unimi.it, lanzarotti@di.unimi.it, petrini@di.unimi.it
[3] Department of Mathematics, Khalifa University of Science and Technology
Al Saada St., PO Box 127788, Abu Dhabi, United Arab Emirates
jianyi.lin@ku.ac.ae

Abstract. In parallel computation domain, graph coloring is widely studied in its own and represents a reference problem for scheduling of parallel tasks. Unfortunately, common graph coloring strategies usually focus on minimizing the number of colors without any concern for the sizes of each color class, thus producing highly skewed color class distributions. However, to guarantee efficiency in parallel computations, but also in other application contexts, it is important to keep the color classes highly balanced in their sizes. In this paper we address this challenging issue for large scale graphs, proposing a fast parallel MCMC heuristic for sparse graphs that randomly generates good balanced colorings provided that a sufficient number of colors are made available. We show its effectiveness through some numerical simulations on random graphs.

Keywords: balanced graph coloring, Markov Chain Monte Carlo method, greedy colorer, parallel algorithms.

1 Introduction

The vertex coloring (or graph coloring) problem is one of the fundamental and most difficult combinatorial problems. Given an undirected graph G, one is looking for an assignment of colors to the vertices of G such that no two adjacent vertices share the same color and the number of different colors used is minimized. Vertex coloring is known to be NP-hard even for planar graphs [7]. Graph coloring has many applications in different research fields. For example, it can be used to register medical or biometric images [16,4] and to find good resource allocation scheme for device-to-device (D2D) communications [22,3], used in modern wireless communication systems [11]. Moreover, graph coloring is extensively used in social networks problem such as Community Identification in Dynamic Social Networks [21], summarization of social networks messages [19], and for

Collective Spammer Detection [6]. A common characteristic of these tasks is that the graphs have very large size, thus requiring a speed up of the traditional greedy sequential coloring heuristics [2], obtained introducing parallelization. In Patter Recognition, also, there are many applications of graph coloring, that are intractable without efficient parallelization: e.g. stochastic pyramids construction [17], graph classification [9], and so on.

In the literature, parallel graph coloring problem has been tackled by several approaches but, at the best of our knowledge, very few address the problem of balancing in parallel manner. One category of them is based on the search of 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 [15]. Examples of such approaches are [13,8]. Another category includes methods that color as many vertices as possible concurrently, tentatively tolerating potential conflicts, while detecting and solving conflicts afterwards (e.g. [1]). 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 [18]. This constraint is very expensive and somehow too stringent for practical applications. *Balanced* coloring relaxes the equitable constraint requiring that any two color class sizes differ by an integer l greater than 1. Few approaches have been proposed to tackle Balanced graph coloring (e.g. [20,14]). 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 [12] 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.

In this article we present an algorithm based on MCMC method producing balanced graph coloring in a parallel way. The main contribution is the introduction of a proposal distribution, independently for each vertex, that promotes overall balancing objectives. The key computational property is that the generation of colorings with such distributions can be carried out on parallel computational models. The technique is tested on large random graphs showing experimentally its effectiveness.

The remainder of the paper is organized as follows: Section 2 describes the proposed algorithm and in Section 3 we prove quantitatively, by some experimental results, the effectiveness of our method.

2 Parallel MCMC Sampling

2.1 Notations

Let $G = \langle V, E\rangle$ be a simple undirected graph of $n = |V|$ vertices and $[k] = \{1, \ldots, k\}$ be a set of colors used to label the vertices. A k-coloring is an assignment $c : V \to [k]$ such that $c = (c(1), \ldots, c(n)) \in [k]^n$ 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. The neighborhood of v induces the partition $\{c_{\mathcal{N}(v)}, \bar{c}_{\mathcal{N}(v)}\}$ of $[k]$ for which $h_v(c) = |c_{\mathcal{N}(v)}|$ and $\bar{h}_v(c) = |\bar{c}_{\mathcal{N}(v)}|$ denote their cardinality. 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

Monte Carlo estimation methods [23] are a broad class of statistical sampling techniques based on the idea of estimating an unknown quantity with averaging over a large set of samples. Due to the strong law of large numbers, the estimate is guaranteed to almost surely converge to the unknown quantity. When the iterative sampling scheme is based on the distribution of a Markov Chain it is termed Markov Chain Monte Carlo (MCMC). Due to the lack of space here, we refer to [23] for a comprehensive introduction to the topic.

Our objective is to define a 1^{st}-order ergodic Markov chain $(C_i)_{i=1}^{\infty}$ consisting in a sequence of k-colorings of a simple undirected graph $G = \langle V, E\rangle$ with $n = |V|$ nodes, whose stationary distribution π strongly depends on the set of conflicts (edges with endpoints sharing the same color) involved in each coloring $c \in [k]^n$. A natural target stationary distribution for the Markov chain is the *Gibbs distribution*, expressed in the form

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

where $\#(c) : [k]^n \to \mathbb{N}$ counts the number of conflicts of the coloring c.

The choice of Gibbs distribution (1) turns out to be useful since, when β is sufficiently large, it is close uniformly and with exponential rate to the uniform distribution over the *proper* colorings, which is our desired working set. 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 established Metropolis-Hastings algorithm [10]. This technique prescribes 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 [23]: first a candidate coloring C' is generated according to a suitable proposal probability $r(c, c') := \mathbb{P}(C' = c' \mid C = c)$, then the proposal C' is accepted effectively as successive coloring C^* according to the acceptance ratio $\alpha(c, c')$:

$$\mathbb{P}(C^* = c' \mid C = c) = \alpha(c, c') := \min\left\{\frac{\pi(c')r(c', c)}{\pi(c)r(c, c')}, 1\right\}, \qquad \text{where } c' \neq c$$

while the old coloring $C = c$ is retained with remaining probability $1 - \alpha(c, c')$.

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 Generation of proposal color

Here we specify the proposal probability in algorithmic vein 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 *available* colors $\bar{c}_{\mathcal{N}(v)}$ following a nearly uniform distribution of $C'_v = j$ given c:

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

The rationale behind such definition is that we want with high probability to generate a color equally likely among those available, 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 available, in order to widen the search space. In the clearly rare case of no available colors, the algorithm maintains the old color c_v with high probability, $1 - \varepsilon(n - 1)$, and selects another color with small probability ε.

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

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

So far, we have presented the elements for determining the forward probability $r(c, c')$ in (2). Indeed, from the discussion above one can derive the following

Proposition 1. *The proposal probability of each node v is*

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

On the other hand, 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 $\alpha(c, c')$.

2.4 Algorithm

We can now describe more formally the procedural steps corresponding to our method providing Algorithm 1. As for the convergence properties of the proposed algorithm, since we cannot guarantee that the number of conflicts $\#(C)$ strictly decreases at each iteration, due to the randomness of the MCMC methodology, we are able to give a characterization of the convergence in stochastic terms as in the following proposition, which is stated without proof for lack of space.

Algorithm 1 Parallel MCMC Balanced 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 $C \in [k]^n$

$C :=$ some initial arbitrary coloring $\in [k]^n$
while $\#(C) > 0$ **do**
  for each $v \in V$ **in parallel do**
    Calculate $C_{\mathcal{N}(C)}$ and $h_v(C) := |C_{\mathcal{N}(C)}|$
    Compute $\mathbb{P}(c'_v \mid C)$ according to the rule in Prop. 1
    Generate proposal color C'_v with distribution $\mathbb{P}(c'_v \mid C)$
  Proposed coloring $C' := (C'_1, C'_2, ..., C_n)$
  Compute forward and backward probabilities $r(C, C')$, $r(C', C)$
  $\alpha(C, C') := \min\{\frac{r(C', C)}{r(C, C')} e^{-\beta(\#(C') - \#(C))}, 1\}$
  Accept proposed coloring, $C := C'$, with probability $\alpha(C, C')$
return C

Proposition 2. *Let C^t and C^{t+1} be the random coloring in iteration t and $t+1$, respectively, of Algorithm 1 on G. Provided a number of colors $k \geq \Delta(G)+1$, it holds the expectation inequality:*

$$\mathbb{E}[\#(C^{t+1})] < \mathbb{E}[\#(C^t)].$$

It follows that, the number of colors $\#(C)$ in the algorithm converges in probability to 0, i.e. $\lim_{t\to\infty} \mathbb{P}(\#(C^t) < \theta) = 1 \ \forall \theta > 0$.

3 Numerical simulations

In this section we report some numerical simulation results of the parallel MCMC method exploiting the Erdős-Rènyi graph (ER) model [5]. This model is widely used to generate random graphs and has some valuable properties to leverage in order to asses the behaviour of the proposed coloring strategy both for fixed graph sizes and asymptotically.

In the ER model $G(n,p)$, a n-vertex graph is constructed by connecting vertices randomly and including each edge with probability p independently from every other edge. Equivalently, the probability that a vertex v has degree k is Binomial, i.e. $\mathbb{P}(\deg(v) = k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$, with expected value $E[\deg(v)] = (n-1)p$. As n goes to infinity, the probability that a graph in $G(n, 2\ln(n)/n)$ is connected, tends to one. Another relevant property of ER graphs is the edge density which is a random variable with expectation exactly equal to the background probability p.

The role that ER model plays in this picture is important because it ensures the possibility to provide graphs with certain properties, giving us at same time an effective and sound algorithmic procedure to compute them in practice.

To compare our MCMC strategy with other techniques designed for the same purpose, we choose a fast parallel greedy algorithm inspired by Luby's work [15]. Luby has described a greedy parallel strategy to find a maximal independent set (MIS) of vertices (i.e. a subset of vertices such that no two vertices are neighbors) in undirected graphs. Consequently, given that any MIS can be colored in parallel, a greedy graph coloring strategy could be defined by repeatedly finding the largest MIS on subgraphs gradually resulting from pruning previous recovered MIS.

Clearly, the Luby inspired colorer is not meant for balanced graph coloring problem. However, we use it for two reasons: on one hand just for sake of comparison with a simple scheme graph colorer, on the other hand to empirically show that the two algorithms have comparable computational times on sparse graphs.

3.1 CUDA parallel implementations

We developed a fast parallel implementation of both the MCMC and the Luby-greedy coloring algorithms, called respectively MCMC-GPU and Luby-GPU,

using the NVIDIA CUDA programming paradigm. NVIDIA GPU processors feature up to 5000 processing cores, hence a very large number of processing threads can be scheduled and executed concurrently in a shared memory model. Thanks to this, parallelization in both MCMC-GPU and Luby-GPU occurs at vertex level, i.e. a thread is assigned to each vertex of the graph which is therefore processed concurrently to every other vertex.

MCMC-GPU implementation closely follows Algorithm 1: during the iterations, each vertex v is assigned to a processing thread which evaluates both $\mathbb{P}(c'_v \mid c)$ and $\mathbb{P}(c_v \mid c')$ (the forward and backward probabilities) and draws the new color c_v accordingly. Thread synchronization occurs only at the end of each iteration, where the total number of conflicts and the rejection factor $\alpha(c, c')$ of the new coloring have to be evaluated. In MCMC-GPU the algorithm is executed until a proper k-coloring is found, or the maximum number of allowed iterations is reached.

Also in Luby-GPU parallelization occurs at vertex level: each vertex is assigned to a thread that evaluates its status (free or not-free) and randomly adds itself to the current MIS if and only if it does not generate any conflict. Synchronization is more invasive, since it has to be performed in every stage of the MIS assembly.

3.2 Performances on ER graphs

Here we report some preliminary experimental results on ER graphs comparing the MCMC-GPU and Luby-GPU algorithms running on NVIDIA GPU devices.

We aim at measuring the quality of balancing in the color class sizes produced in particular by MCMC-GPU, which works by improving the balancing of an existing (improper) coloring moving vertices from one color class to another on the basis of random choices, as highlighted by the proposed distribution over colors given in Section 2.3. Let us first introduce a measure of deviation of a coloring $c \in [k]^n$ from a perfectly balanced coloring where each color class j has size $n_j = n/k$, for each $j \in [k]$. This can be quantified by defining $\gamma_{n,k}(j) = |f_j - n/k|$ which represents the target to be minimized in respect of which our heuristic will perform local random arrangements. An overall measure of balancing quality closely related to the standard deviation of the color class sizes can be then defined as

$$\Gamma_{n,k}(c) = \left(\frac{1}{k} \sum_{j=1}^{k} \gamma_{n,k}^2(j) \right)^{1/2} \tag{3}$$

called *unbalancing index* hereafter. Clearly, a coloring c is perfectly balanced if $\Gamma_{n,k}(c) = 0$.

A demonstration on how MCMC-GPU and Luby-GPU perform in terms of color balancing is given in Fig. 1, where the curves represent the unbalancing index (3). More precisely, the graphics relate to average amounts achieved on ER graphs of various size and densities 0.1% and 0.5% respectively. To capture a wide scale of ER graphs, once the density p has been fixed, we varied the graph

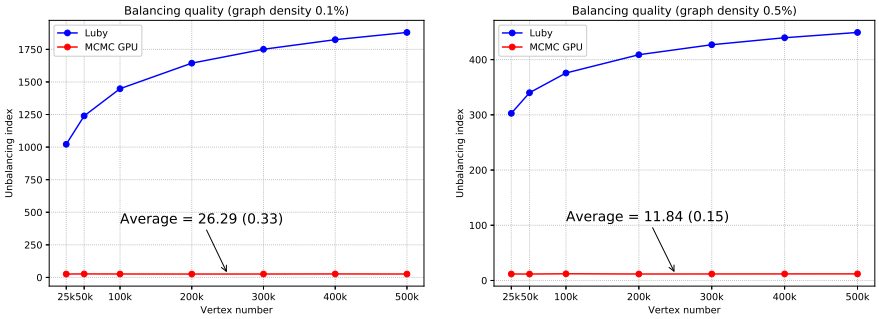

Fig. 1. Average unbalancing index achieved by MCMC-GPU and Luby-GPU on ER graphs of various size and densities 0.1% and 0.5% respectively.

size n up to 500K vertices, averaging over 10 trials for each pair (n, p) to reach satisfactory confidence level.

Concerning the number of colors k used by MCMC-GPU, assuming a regular structure of the generated graphs we fix $k = \lceil np \rceil$ corresponding to the expected vertex degree, which anyway assures the existence of proper coloring with high probability. Note that, under this setting a coloring c for a graph in $G(n, p)$ has balancing index

$$\Gamma_{n,\lceil np \rceil}(c) \approx \left(\frac{1}{np} \sum_{j=1}^{k} \gamma_{n,k}^2(j) \right)^{1/2} .$$

As can be noticed in the plots, MCMC-GPU not only outperforms Luby-GPU, but also provides invariant unbalancing index with respect to the graph sizes, being $\Gamma_{n,\lceil np \rceil}(c) \approx$ constant (with very low standard deviation), for all n in the range 25K $\div$ 500K. In spite of the limited number of experiments, as a general trend we have that the sum of within-class deviations $\gamma_{n,k}^2(j)$ roughly grows linearly with the number of vertices, i.e. $\sum_{j=1}^{k} \gamma_{n,k}^2(j) \approx pc_p n$, where c_p is a constant depending on the density p. For instance, in Fig. 1, the constants $c_p = 26.29$ and $c_p = 11.84$ are shown for $p = 0.1\%$ and $p = 0.5\%$ respectively.

With regard to the computational times of the conducted experiments, their averages are reported in Fig. 2. Whereas we can notice very high speedups (up to 20) between sequential and parallel MCMC implementations, the times spent by MCMC-GPU and Luby-GPU remain comparable (they turn in favor of Luby-GPU only for 500K).

A second experiment is aimed at studying the balancing quality achieved when varying both the graph density and the number of colors made available to MCMC-GPU. In particular, here we set the graph density in terms of vertex degree $d = \lceil np \rceil$, with d falling in the range 100 $\div$ 350, while the number $k = r \lceil np \rceil$ of colors is scaled down by a factor $r \in (0, 1)$ ranging from 0.5 and 1. Average values of the unbalancing index over colorings carried out by the two algorithms are plotted in Fig. 3 (note that the ratio between the scales of the two graphs is about 33).

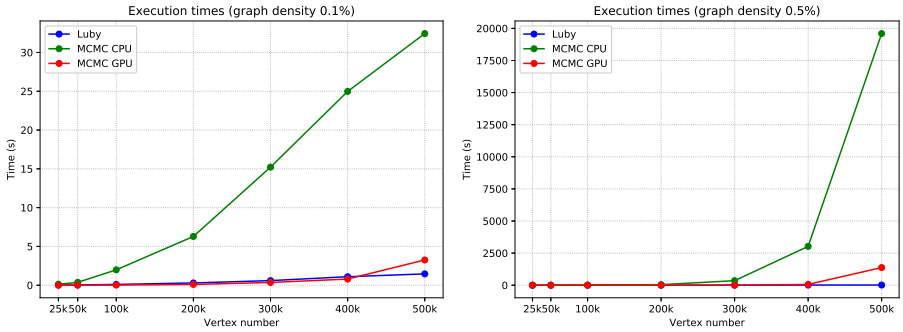

Fig. 2. Average execution times of Luby-GPU, MCMC-CPU and MCMC-GPU on ER graphs of various size and densities 0.1% and 0.5% respectively.

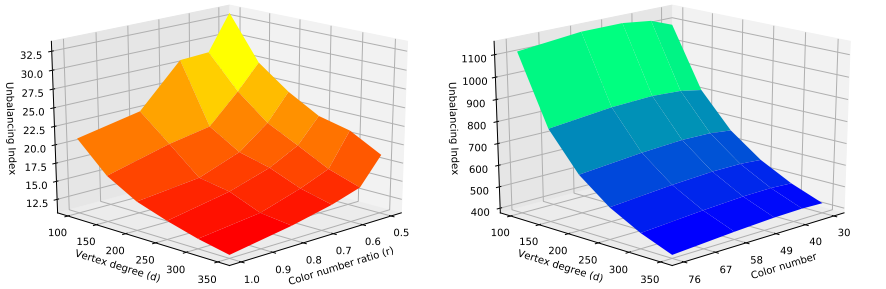

Fig. 3. Average unbalancing index achieved by MCMC-GPU (left) and Luby-GPU (right) on ER graphs varying both vertex degrees $d = \lceil np \rceil$ and color number k.

4 Conclusions

In this paper we have proposed a new parallel algorithm for graph coloring problem based on Markov Chain Monte Carlo techniques. The main goal of this new method is to produce balanced solutions, which is a direction not much explored yet in the literature. Experiments show the effectiveness of the approach on random graphs. Given the encouraging results shown in this article, further investigations are deserved. In particular we intend to generalize the model in order to be able to optimize several forms of balancing, possibly redefined at each iteration of parallel coloring; furthermore, we will study a theoretical analysis of the model, and finally we will extend the experiments testing the algorithm on different graph typologies and comparing it with other parallel approaches.

References

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

2. Coleman, T.F., Moré, J.J.: Estimation of sparse Jacobian matrices and graph coloring problems. SIAM Journal on Numerical Analysis **20**(1), 187–209 (1983)
3. Comi, P., Crosta, P.S., Beccari, M., Paglierani, P., Grossi, G., Pedersini, F., Petrini, A.: Hardware-accelerated high-resolution video coding in virtual network functions. In: Eu. Conf. on Network and Communication. pp. 32–36 (2016)
4. Cuculo, V., Lanzarotti, R., Boccignone, G.: Using sparse coding for landmark localization in facial expressions. In: 5th European Workshop on Visual Information Processing, EUVIP 2014. pp. 1–6. IEEE (2014)
5. Erdős, P., Rènyi, A.: On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci **5**, 17–61 (1960)
6. Fakhraei, S., Foulds, J., Shashanka, M., Getoor, L.: Collective spammer detection in evolving multi-relational social networks. In: Proc. 21th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining. pp. 1769–1778. ACM (2015)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)
8. Gjertsen, R.K., Jones, M.T., Plassmann, P.E.: Parallel heuristics for improved, balanced graph colorings. J. Par. and Dist. Comput **37**, 171–186 (1996)
9. Gómez, D., Montero, J., Yáñez, J.: A coloring fuzzy graph approach for image classification. Information Sciences **176**(24), 3645–3657 (2006)
10. Hastings, W.: Monte carlo sampling methods using markov chains and their applications. Biometrika **57**(1), 97–109 (1970)
11. Janis, P., Chia-Hao, Y., Doppler, K., Ribeiro, C., Wijting, C., Klaus, H., Tirkkonen, O., Koivunen, V.: Device-to-device communication underlaying cellular communications systems. Int. Jour. of Comm., Network and System Sciences **2-3** (2009)
12. Jerrum, M.R.: A very simple algorithm for estimating the number of k-colourings of a low-degree graph. Random Structures and Algorithms **7**, 157–165 (1995)
13. Jones, M.T., Plassmann, P.E.: A parallel graph coloring heuristic. SIAM J. Sci. Comput. **14**, 654–669 (1992)
14. Lu, H., Halappanavar, M., Chavarria-Miranda, D., Gebremedhin, A., Kalyanaraman, A.: Balanced coloring for parallel computing applications. In: Proc. IEEE 29th Int. Parallel and Distributed Processing Symposium. pp. 7–16 (2015)
15. Luby, M.: A simple parallel algorithm for the maximal independent set problem. In: Proc. 17th Annual ACM Symposium on Theory of Computing. pp. 1–10 (1985)
16. Lupaşcu, C.A., Tegolo, D., Bellavia, F., Valenti, C.: 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. pp. 453–456. IEEE (2013)
17. Meer, P.: Stochastic image pyramids. CVGIP **45**(3), 269–294 (1989)
18. Meyer, W.: Equitable coloring. Amer. Math. Monthly **80**, 920–922 (1973)
19. Mosa, M.A., Hamouda, A., Marei, M.: Graph coloring and aco based summarization for social networks. Expert Systems with Applications **74**, 115–126 (2017)
20. Robert, J., Gjertsen, K., Jones, M.T., Plassmann, P.: Parallel heuristics for improved, balanced graph colorings. Journal of Parallel and Distributed Computing **37**, 171–186 (1996)
21. Tantipathananandh, C., Berger-Wolf, T., Kempe, D.: A framework for community identification in dynamic social networks. In: Proc. 13th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining. pp. 717–726. ACM (2007)
22. Tsolkas, D., Liotou, E., Passas, N., Merakos, L.: A graph-coloring secondary resource allocation for d2d communications in lte networks. In: IEEE 17th Int. Workshop on Computer Aided Modeling and Design (CAMAD). pp. 56–60. IEEE (2012)
23. Voss, J.: An introduction to statistical computing: a simulation-based approach. John Wiley & Sons (2013)