Brief Announcement: Fast and Simple Node Coloring in the SINR Model

Fabian Fuchs
Institute of Theoretical Informatics, KIT
Karlsruhe, Germany
fabian.fuchs@kit.edu

ABSTRACT

We present two distributed node coloring algorithms operating in the Signal-to-interference-and-noise-ratio (SINR) model. The first algorithm is very simple and achieves a (4Δ)-coloring in $\mathcal{O}(\Delta \log n)$ time slots. The results of our experimental evaluation show that the algorithm is extremely fast. Combined with a new color reduction scheme, the algorithm computes a $(\Delta+1)$-coloring in $\mathcal{O}(\Delta \log n)$ time. This improves on current distributed coloring algorithms for the SINR model either in terms of the number of colors or runtime, and matches the asymptotical runtime of one round of local broadcasting, which can be seen as a lower bound.

Categories and Subject Descriptors

F.2.m [**Analysis of Algorithms**]: Miscellaneous; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*distributed networks*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*

Keywords

Algorithms; Distributed Node Coloring; SINR Model; Wireless Ad Hoc Networks; Experiments

1. INTRODUCTION

Distributed node coloring algorithms can be used to make communication in wireless (ad-hoc) networks more efficient by establishing coordinated medium access, for example by using Time Division Multiple Access (TDMA). We use the geometric Signal-to-interference-and-noise-ratio (SINR) model of interference, which is widely considered to be realistic. Thus, many algorithmic works considered this model in the last decade. Communication of distributed algorithms in the SINR model is often based on probabilistic medium access. This yields the best solutions to the local broadcasting problem (see [1] and the references in [2]), in which all nodes in the network must transmit one message to all their neighbors. In distributed node coloring algorithms, one aims for $\Delta+1$ colors, as this can be achieved for any communication graph, and minimizing the number of colors beyond this level is hard even for the centralized case. There are currently two algorithms on a pareto front: The Yu *et al.* algorithm [3] computes a $(\Delta+1)$-coloring in $\mathcal{O}(\Delta \log n + \log^2 n)$ time slots, while the MW-coloring algorithm [4, 5] executed in the SINR model establishes an $\mathcal{O}(\Delta)$-coloring in $\mathcal{O}(\Delta \log n)$ time slots.

In this announcement, we briefly state recent results regarding more efficient distributed node coloring algorithms in the SINR model by Fuchs and Prutkin [6,7]. They adapted simple and well-known coloring algorithms from the message-passing model to the SINR model. The first algorithm runs in $\mathcal{O}(\Delta \log n)$ time slots and computes a valid (4Δ)-coloring. The results on an experimental evaluation show that the algorithm is fast, even compared to a local broadcast. Additionally, it can be combined with a new color reduction scheme, which reduces a given d-coloring in $\mathcal{O}(d \log n)$ time slots to a $(\Delta+1)$-coloring, yielding a $(\Delta+1)$-coloring that can be computed in $\mathcal{O}(\Delta \log n)$ time. This outperforms MW-coloring (under the same assumptions) by using less colors, and the Yu *et al.* algorithm by requiring less time slots. Also, this essentially closes the field—unless faster local broadcast algorithms emerge—as the goal of $\Delta+1$ colors is achieved in a runtime that matches current local broadcasting algorithm.

1.1 Preliminaries

We consider a network of n nodes, and use the SINR model to decide whether a transmission from a node v can be decoded at a node u. The transmission is *feasible* at u iff $\frac{P/\operatorname{dist}(v,u)^\alpha}{\sum_{w \in I} P/\operatorname{dist}(w,u)^\alpha + N} > \beta$, where P is the transmission power, $\operatorname{dist}(u,v)$ is the Euclidean distance from u to v, I the set of nodes transmitting simultaneously to v, α the attenuation coefficient depending on the environment, β a hardware-dependent threshold, and N the environmental noise. We define the broadcasting range of each node based on these constants, which induces a communication graph $G=(V,E)$, and more specifically a set of neighbors N_v for each node v. The maximum number of neighbors (max. degree) is denoted by Δ. Two nodes are called *independent*, if they are not neighbors. A set of nodes is independent if no two nodes are neighbors. The network is *colored* with d colors if the nodes are partitioned in d sets. The coloring is *valid*, if each set is independent. We say that a transmission of v is *successful*, if it can be received by all neighbors of v. Apart from classical local broadcasting, which achieves successful transmission with high probability (w.h.p.–with probability

PODC'15, July 21–23, 2015, Donostia-San Sebastián, Spain.
ACM 978-1-4503-3617-8 /15/07.
http://dx.doi.org/10.1145/2767386.2767445.

at least $1-\frac{1}{n}$) in $\mathcal{O}(\Delta \log n)$ time using a transmission probability $p_1 = \frac{1}{\mathcal{O}(\Delta)}$, we use two straight-forward extensions to this result in our coloring algorithms. The extensions are stated as lemmas 1 and 2. Our assumptions match those of local broadcasting with known Δ, i.e. we assume Δ, α, β, N, and a polynomial estimate of n to be given.

LEMMA 1. *Let all nodes transmit with transmission probability $p_1 = \frac{1}{\mathcal{O}(\Delta)}$, then a transmission from a node v is successful within $\mathcal{O}(\Delta)$ time slots with probability at least $\frac{3}{4}$.*

LEMMA 2. *Let $\mathcal{I}$ be a set of independent nodes transmitting with probability $p_2 = \frac{1}{\mathcal{O}(1)}$ and the remaining nodes with probability p_1. A transmission from node $v \in \mathcal{I}$ is successful within $\mathcal{O}(\log n)$ time slots w.h.p.*

Both, local broadcasting and our extensions, guarantee that the transmission can successfully be decoded by nodes in the broadcasting range. However, the transmission might also be received by nodes in the transmission range, which is larger than the broadcasting range to enable spacial reuse. Thus, to guarantee a $(\Delta + 1)$-coloring, we increase Δ by a constant factor or assume that the nodes discard transmissions from outside the broadcasting range (e.g. based on the received signal strength). Note that the maximum degree Δ is defined as in previous $\Delta + 1$ coloring algorithms [3].

2. RESULTS

Let us first consider RAND4ΔCOLORING (Algorithm 1), which is based on a folklore algorithm for the message-passing model: Each node repeatedly transmits its color and selects a new color once a conflict is detected. Our main idea is to speed up the algorithm by restricting the algorithm to transmissions that are successful with constant probability, which requires only $\mathcal{O}(\Delta)$ time, cf. Lemma 1, instead of with high probability, which would require $\mathcal{O}(\Delta \log n)$ time. This allows to run the algorithm in $\mathcal{O}(\Delta \log n)$ time in the SINR model instead of $\mathcal{O}(\Delta \log^2 n)$ time slots for a simple simulation of the algorithm. However, the improved runtime comes at the price that the introduced uncertainty must be handled in the analysis.

Algorithm 1: RAND4ΔCOLORING for node v

1 **for** $t \leftarrow 0; t \leq \mathcal{O}(\ln n); t \leftarrow t+1$ **do** // each: one phase
2   Transmit c_v^t with prob. p for $\mathcal{O}(\Delta)$ time slots;
3   **foreach** *received color* c_w^t *from neighbor* $w \in N_v$ **do**
4         $F_v \leftarrow F_v \backslash \{c_w^t\}$
5   **if** $c_v^t \notin F_v$ **then** $c_v^{t+1} \leftarrow [4\Delta]$.rand(); // conflict
6   **else** $c_v^{t+1} \leftarrow c_v^t$; // otherwise, keep color
7     $F_v \leftarrow [4\Delta]$;

The algorithm itself is divided in several phases. During each phase, the considered node v transmits its current color and receives colors of some neighbors. By removing those colors from the set of free colors F_v, they are marked as taken. At the end of each phase, v evaluates whether it detected a conflict with a neighbor (Line 5), and resets the color if so. Otherwise it keeps the color and the set of free colors is reset for the next phase. We can show that

THEOREM 3. *Algorithm 1 computes a valid (4Δ)-coloring in $\mathcal{O}(\Delta \log n)$ time w.h.p.*

This holds for the synchronous case without restriction, however, for the asynchronous case we need to account for conflicts introduced by nodes waking-up late. Thus, the result holds for nodes with a stable $\log n$ neighborhood, while the bound cannot be guaranteed if nodes in the $\log n$ neighborhood wake-up in the meantime. Once the nodes in the $\log n$ neighborhood are awake, the bound holds.

2.1 Color Reduction

The COLORREDUCTION algorithm is based on the simple observation that an independent set of nodes (or even a constant number of independent sets) can transmit with increased probability, allowing a local broadcast of these independent sets in $\mathcal{O}(\log n)$ time w.h.p., as stated in Lemma 2. Thus, once we have computed a valid coloring, this can be used to increase the communication efficiency. As the asynchronous case requires considerably more synchronization effort, let us first describe the case of a synchronous, simultaneous start (see Algorithm 2; for pseudocode of the asynchronous version cf. [6]).

Given a valid d-coloring of the network and assume each color is an integer from the set $[d]$. Divide the time in epochs of length $\mathcal{O}(\log n)$ and let the nodes of color i transmit in epoch i. This ensures that during an epoch nodes of only one color transmit, hence those nodes can select a free color from $[\Delta]$ and inform their neighbors about their selection. This does not lead to a conflict w.h.p., as nodes from previous epochs informed their neighbors, and no two neighbors are active at the same time due to the valid coloring. Thus, after d epochs the networks is colored with $\Delta + 1$ colors w.h.p.

Algorithm 2: COLORREDUCTION for v; color c_v

$F_v \leftarrow [\Delta]$;
for $i \leftarrow 0; i \leq d; i \leftarrow i+1$ **do** // each: one epoch
    if $c_v = i$ **then** transmit $c_v \leftarrow F_v$.rand() with ;
    probability p_2 for $\mathcal{O}(\log n)$ time slots;
    else // wait
        listen for $\mathcal{O}(\log n)$ time slots;
        foreach *received color* c_w **do** $F_v \leftarrow F_v \backslash \{c_w\}$;

Let us now briefly consider the asynchronous case. In this case, nodes cannot implicitly decide on a common start of the schedule as done in the synchronous case. Thus, without global synchronization we cannot guarantee that only the nodes of a constant number of colors transmit. However, it is sufficient if this property holds locally - thus in each neighborhood only the nodes of a constant number of colors are allowed to transmit. We can achieve this by computing two layers of maximal independent sets (MIS). Nodes of the first layer MIS act as leaders and decide on a schedule. All other nodes select a leader in their neighborhood and follow its schedule. If all nodes follow their leaders schedule, it holds for each node that at most a constant number of neighbors may be active at the same time. This is sufficient to apply a variant of Lemma 2, as the active nodes can be grouped to a constant number of independent sets. Executing an MIS algorithm amongst the active nodes reduces the set to an actual MIS, which allows the nodes in the MIS to select a color from $[\Delta]$ and communicate the selected color to its neighbors successfully without a conflict. Let us consider the runtime: The first layer of MIS runs in $\mathcal{O}(\Delta \log n)$ time slots, for example by using a known MIS algorithm [5]. Selecting a leader and communicating the schedule requires the same asymptotic time. By reducing the number of active nodes to a constant, we can modify the MIS algorithm to run in $\mathcal{O}(\log n)$ time slots. Using an epoch of length $\mathcal{O}(\log n)$ suffices to ensure that each active node is in the MIS once after a constant number of MIS executions, as there are a constant number of active nodes in each epoch. As we assume a valid d-coloring to be given, d epochs, and thus $\mathcal{O}(d \log n)$ time slots are sufficient to validly color the network with $\Delta + 1$ colors.

2.2 Experiments

We use the Sinalgo simulator for our experiments (cf. [7] for more details). The nodes are deployed uniformly at random on an area of 1000×1000 meters. We use a uniform transmission power of 1 and SINR constants $\alpha = 4$, $\beta = 10$, $\mathrm{N} = 1^{-9}$, resulting in a broadcasting range of 84 meters. Algorithm 1 uses a phase length equivalent to 10 time slots, the nodes start asynchronous but within the first phase. Our results are robust regarding those parameters, and are obtained using 100 runs each. Figure 1 shows that it takes significantly less time to compute a valid (4Δ)-coloring for the network than for each node to transmit a message to its neighbors (i.e. finish one round of local broadcasting).

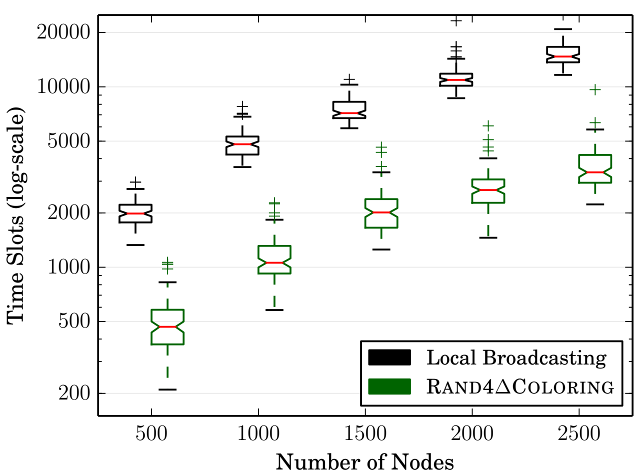

Figure 1: Required time to finish one round of local broadcasting, and computing a valid (4Δ)-coloring using Algorithm 1. Note the log-scaled y-axis.

We consider mobility of the nodes according to the random direction mobility model with variable node speed in Fig. 2. Note that in such a network some conflicts are inevitable due to the movement of nodes. As mobility in only available for synchronous execution, this simulation uses synchronous rounds[1]. We observe that RAND4ΔCOLORING is relatively robust. Even under moderate mobility values of 1 meter per time slot, more than 90% of the nodes have a valid color. We note that other current distributed node coloring algorithms for the SINR model are not viable in

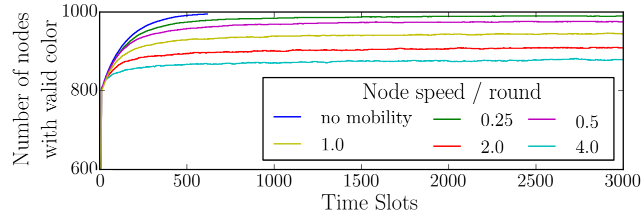

Figure 2: Consider a network of 1000 mobile nodes. We measure the number of nodes that have a valid color in each time step. The performance decreases with increasing mobility.

the mobile setting, as they do not resolve conflicts, which inevitably emerge using mobile nodes.

3. CONCLUSION

We proposed two very simple node coloring algorithms for the SINR model, which are based on well-known message-passing algorithms. The first algorithm is very simple and computes a (4Δ)-coloring in $\mathcal{O}(\Delta \log n)$ time slots. Our experiments validated that RAND4ΔCOLORING is extremely fast, even compared to a simple local broadcast. Our second algorithm is based on a color reduction scheme, which has an almost trivial synchronous implementation in the SINR model. For the asynchronous case, the algorithm is based on two layers of MIS, which allow the nodes to compute a $(\Delta+1)$-coloring in $\mathcal{O}(\Delta \log n)$ time slots based on an $\mathcal{O}(\Delta)$-coloring. This results in an $(\Delta + 1)$-coloring algorithm in $\mathcal{O}(\Delta \log n)$ time slots, which is essentially optimal.

4. ACKNOWLEDGEMENTS:

We thank Magnús M. Halldórsson for helpful discussions on an early stage of this work, and the German Research Foundation (DFG), which supported this work within the Research Training Group GRK 1194 "Self-organizing Sensor-Actuator Networks".

5. REFERENCES

[1] O. Goussevskaia, T. Moscibroda, and R. Wattenhofer, "Local Broadcasting in the Physical Interference Model," in *Proc. 5th DialM-POMC*. ACM Press, 2008, pp. 35–44.

[2] M. M. Halldórsson and P. Mitra, "Towards tight bounds for local broadcasting," in *Proc. 8th FOMC*. ACM Press, July 2012.

[3] D. Yu, Y. Wang, Q.-S. Hua, and F. C. M. Lau, "Distributed $(\Delta + 1)$ Coloring in the Physical Model," in *ALGOSENSORS*, ser. LNCS, vol. 7111. Springer, 2011, pp. 145–160.

[4] T. Moscibroda and M. Wattenhofer, "Coloring unstructured radio networks," *J. Distr. Comp.*, vol. 21, no. 4, pp. 271–284, 2008.

[5] B. Derbel and E.-G. Talbi, "Distributed Node Coloring in the SINR Model," in *ICDCS*. IEEE, 2010, pp. 708–717.

[6] F. Fuchs and R. Prutkin, "Simple distributed $\Delta + 1$ coloring in the SINR model," 2015. [Online]. Available: http://arxiv.org/abs/1502.02426

[7] F. Fuchs, "On asynchronous node coloring in the SINR model," 2015, manuscript. [Online]. Available: http://i11www.iti.kit.edu/f-oancs-15.pdf

[1] The synchronized variant without mobility finishes within 600 slots in Fig. 2 compared to about 1000 slots for the asynchronous setting in Fig. 1. This exposes the gap between slotted vs. unslotted transmissions, see e.g. [1].