On Asynchronous Node Coloring in the SINR Model

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

Abstract

In this work we extend the analysis of Fuchs and Prutkin [1] towards the asynchronous case of the RandCDeltaColoring algorithm. This results in an 4Δ coloring algorithm that runs in $\mathcal{O}(\Delta)$ time slots in the Signal-to-interference-and-noise (SINR) model. Additionally we show in a simulation that the constants hidden in the $\mathcal{O}$-notation are small, resulting in an algorithm that is extremely fast, even if compared to one round of local broadcasting.

Note that this material extends a brief announcement submitted to PODC'15.

1 Introduction

Distributed node coloring algorithms can be used to make communication in wireless (ad-hoc) networks more efficient by establishing coordinated medium access, as for example 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 (see [2–6]) to the local broadcasting problem, 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 is hard even for the centralized case. There are currently two algorithms on a pareto front: The Yu *et. al* algorithm [7] computes a $\Delta + 1$ coloring in $\mathcal{O}(\Delta \log n + \log^2 n)$ time slots[1], while the MW-coloring algorithm [8] executed in the SINR model establishes an $\mathcal{O}(\Delta)$ coloring in $\mathcal{O}(\Delta \log n)$ time slots.

For related work regarding the theoretical part of this work, we refer to [1]. Regarding the experimental evaluation, we are the first to evaluate distributed node coloring in the SINR model experimentally to the best of our knowledge. Local broadcasting has been evaluated in [6]. They used an area of 1000×1000, with a similar setting as ours, however a broadcasting range of only 25, leading to an average density of 2 to 10 for 1000 to 5000

[1]Note that time is divided in slots for the analysis, however, our asynchronous algorithm do not require global time slots.

nodes. The increased broadcasting range leads to a considerable higher density in our evaluation.

RandCDeltaColor can be seen as a simple variant of Luby's MIS algorithm [9] in the message-passing model. There, in each round a node tries to select a color, if no neighbor selected the same color the node finalizes the color, and otherwise selects a new color in the next round[2]. Different variants of this algorithm are experimentally evaluated by Finocchi, Panconesi and Silvestri in [10].

Roadmap

In the following section we briefly introduce the notation required for our analysis. In the following Section 3 we recapitulate the main parts of the RandCDeltaColoring algorithm before proving that it computes a valid 4Δ coloring in $\mathcal{O}(\Delta \log n)$ time also in the asynchronous SINR model. In Section 4 we experimentally evaluate the algorithm by implementing it in the Sinalgo network simulator [11], and comparing it to the basic local broadcasting algorithm.

2 Preliminaries

We consider a network of n nodes and use the SINR model to decide whether a transmission from a node v can successfully 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. We use a more general notation to describe the j neighborhood of a node v (including v) by $N_v^j := \{u \in V | u \text{ is in the } j \text{ neighborhood of } v\}$. For brievity we use $N_v^+ = N_v \cup \{v\}$ for N_v^1.

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 prob. 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 a straight-forward extension to this result in the coloring algorithm. The extension is stated as Lemma 1. 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 (cf. [6].

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

[2] Note that both the synchronous and the asynchronous version of RandCDeltaColor are based on this simple idea, however, adapting the algorithm to be efficient in the SINR model requires significant effort.

3 Algorithm

The algorithm considered in this section is exactly the same as described in [1]. To be self contained a pseudocode can be found as Algorithm 1. We shall extend the analysis to the case of asynchronous node wake-up in the following.

The algorithm is based on several phases. Each phase consists of a time interval that fits the time required to communicate with constant probability (cf. Lemma 1. During each phase the node transmits its current color, and evaluates whether a conflict was detected or not just before the phase ends. If a conflict was detected, a new color is randomly selected, and transmitted in the next phase. To argue about phases, we number them, individually for each node and say that time t_v is exactly in-between phase t_v and $t_v + 1$ of node v. Note that evaluation and a potential reset happens within the phase and therefore right before t_v. As we argue about a node v, we shall omit the v if it is clear from the context that we refer to t_v.

Regarding a node v and time t_v, we denote the end of the current phase at node u by $t_u^{(v)}$. Again, we may omit (v) for brevity when arguing about v.

Algorithm 1: AsyncRandColoring for node v

---

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

Given a node v and time t. We use $c\Delta$ colors, thus if a node v resets, the color of a neighbor u of v is randomly selected with probability at most $\frac{1}{c\Delta}$. A union bound over all neighbors yields a probability of $\frac{1}{c}$ that v selects the color of one of its neighbors. Also, nodes fail to transmit to all their neighbors from time t to $t+1$ with probability at most $1/q$.

Let $k = \frac{qc}{c+3q}$ be a constant such that $1/k < 1$ (which holds for $c > 3$ and $q > 4$). Consider the probability that there is a conflict at node v at time $t+1$ given that there was a conflict at v (or v's neighbors u_i) at time t (or t_{u_i}) with probability at most $1/k^t$ $(1/k^{t_u})$. We claim that this probability is at most $1/k^{t+1}$ in the following theorem.

Theorem 2. *Given a node v and a time t such that $t \leq t_u$ for all $u \in N_v^+$. It holds that* $\Pr(\mathrm{cfl}^{t+1}(v) | \forall u \in N_v^+ : \Pr(\mathrm{cfl}^{t_u}(u)) \leq 1/k^t) \leq 1/k^{t+1}$

Proof. If a conflict happens at node v at time $t+1$, this can be attributed to one (ot both) of two situations:

1) There was a conflict at v at time t, which did not get resolved

2) A neighbor of v detected a conflict, reset its color and selected v's color.

We shall prove bounds on the probability of each case separately in lemmas 3 and 4, and prove the theorem based on these results in the following. Let us consider a node v such that $\forall u \in N_v^+ : \mathrm{cfl}^{t_u}(u) \leq 1/k^t$. Then it holds due to lemmas 3 and 4

$$\begin{aligned}
\Pr(\mathrm{cfl}^{t+1}(v)) &\leq \Pr(\mathrm{cfl}^t(v)) \cdot \left(\frac{1}{q} + \frac{2}{c}\right) + \sum_{u \in N_v} \Pr(\mathrm{cfl}^t(u)) \cdot \frac{1}{c\Delta} \\
&\leq \frac{1}{k^t}\left(\frac{1}{q} + \frac{2}{c}\right) + \sum_{u \in N_v} \frac{1}{k^t c\Delta} \leq \frac{1}{k^t}\left(\frac{1}{q} + \frac{2}{c}\right) + \max_{u \in N_v} \frac{1}{k^t c} \\
&\leq \frac{1}{k^t}\left(\frac{1}{q} + \frac{2}{c}\right) + \frac{1}{k^t c} = \frac{1}{k^t} \cdot \left(\frac{1}{q} + \frac{3}{c}\right) = \frac{1}{k^{t+1}}
\end{aligned}$$

We argue for the first inequality in the following by considering the cases that may lead to a conflict at v at time $t{+}1$ separately. For two events $A = \mathrm{cfl}^t(v)$ and $B =$ ”cfl. through neighb.”, we consider

$$\Pr(\mathrm{cfl}^{t+1}(v)) \leq \Pr(\mathrm{cfl}^{t+1}(v) \cap A \cap \neg B) + \Pr(\mathrm{cfl}^{t+1}(v) \cap \neg A \cap \neg B) + \Pr(\mathrm{cfl}^{t+1}(v) \cap B)$$

in the three cases below. Probabilities that are not stated in the respective cases are trivially upper bounded by 1.

$$\begin{aligned}
\Pr(\mathrm{cfl}^{t+1}(v) \cap A \cap \neg B) &\leq \Pr(\mathrm{cfl}^{t+1}(v) | \, \mathrm{cfl}^t(v) \wedge \neg \text{cfl. through neighb.}) \cdot \\
&\qquad \Pr(\mathrm{cfl}^t(v) | \neg \text{cfl. through neighb.}) \cdot \Pr(\mathrm{cfl}^t(v)) \qquad (1) \\
&\leq \Pr(\mathrm{cfl}^{t+1}(v) | \, \mathrm{cfl}^t(v) \wedge \neg \text{cfl. through neighb.}) \cdot \Pr(\mathrm{cfl}^t(v)) \\
&\leq \left(\frac{1}{q} + \frac{2}{c}\right) \cdot \Pr(\mathrm{cfl}^t(v))
\end{aligned}$$

The next case may happen with considerable probability, however, it does not lead to a conflict at v.

$$\Pr(\mathrm{cfl}^{t+1}(v) \cap \neg A \cap \neg B) \leq \Pr(\mathrm{cfl}^{t+1}(v) | \neg \, \mathrm{cfl}^t(v) \wedge \neg \text{cfl. through neighb.}) = 0 \qquad (2)$$

The third case leads directly to a conflict at v, however, it happens only with bounded probability.

$$\begin{aligned}
\Pr(\mathrm{cfl}^{t+1}(v) \cap B) &\leq \Pr(\mathrm{cfl}^{t+1}(v) | \text{cfl. through neighb.}) \cdot \Pr(\text{cfl. through neighb.}) \qquad (3) \\
&\leq \Pr(\text{cfl. through neighb.}) \leq \sum_{u \in N_v} \Pr(\mathrm{cfl}^{t_u}(u)) \cdot \frac{1}{c\Delta}
\end{aligned}$$

This proves the theorem. □

Let us first consider the probability that a conflict occurs given that a conflict exists at v at time t.

Lemma 3. $\Pr(\mathrm{cfl}^{t+1}(v) | \, \mathrm{cfl}^t(v) \wedge \neg$ *confl. through neighbor*$) \leq \left(\frac{1}{q} + \frac{2}{c}\right)$

Proof. Let $X_t(v) := \{u \in N_v | c_u^t = c_v^t\}$ be the set of neighbors conflicting with v at time t. Note that we do not consider nodes in $X_{t+1}(v)$ but not in $X_t(v)$, here, as this case is covered by Lemma 4. Although we can guarantee that neighbors of v received a message with v's color in the interval $[t, t+1]$, we cannot guarantee that a conflicting neighbor u of v detected the conflict with v and reseted, as the phases between neighbors are not synchronized. However, we can argue about the probability that v received a message from u in the interval $[t, t+1]$. We know that during the interval v does not change its color. Thus, v either receives the message (which implies that v detects the conflict) with constant probability, or u detected the conflict in the meantime and selected a new color itself. Let us consider these cases that might lead to a conflict at v at time $t+1$ in the following:

a) at least one node $u \in X_t(v)$ does not reset until $t+1$, and v does not detect the conflict.

b) all nodes in $X_t(v)$ reset before $t+1$, but at least one of them selects v's color.

c) v detects the conflict, resets its color but selects the same color as one of its neighbors.

Case a) can be seen as the worst case, as a conflict is not detected and continues to the next round. However, luckily this happens with bounded probability. The case implies that at least one node attempted to transmit during the whole interval $[t, t+1]$. The probability that this transmission is not received by v is at most $1/12$.

In order to resolve conflicts, one of the remaining cases must happen. In case b), we cannot prove bounds on the probabilities whether a transmission was successful or not. But as all nodes in $X_t(v)$ reseted during the interval $[t, t+1]$, we know that the nodes $u \in X_t(v)$ selected a new random color, which is the same as v's color with probability at most $1/c\Delta$. A union bound implies that the conflict probability is $1/c$ for this case.

Finally, case c) happens with reasonable probability (for which do not have an upper bound, but a lower bound of $q - 1/q$ for each node in $X_t(v)$). The probability that a conflict persists to the next phase is at most $1/c$, as this bounds the probability that v selects the same color as v (regardless of whether the neighbors of v reseted themselves or not).

Overall, this bounds the probability for this case by at most $\left(\frac{1}{q} + \frac{2}{c}\right)$ □

Let us now consider the case that the conflict is introduced by a reseting neighbor in the interval $[t, t+1]$.

Lemma 4. Pr(cfl. *through neighb.*) $\leq \sum_{u \in N_v} \frac{1}{k^{t_u}} \cdot \frac{1}{c\Delta}$

Proof. We consider the case that the conflict at v was introduced regardless of whether there was a conflict at v at time t. Thus, $X_t(v)$ might not be empty, however, we consider only the probability of the case that the conflict was introduced through a neighbor's reset. The probability for a neighbor u of v to reset during the interval $[t, t+1]$ is bounded by $1/k^{t_u}$ (as this bounds the probability for a conflict at u at the end of the phase that ends during the interval). Note that such a conflict may be with v, or another neighbor of u. If the conflict is detected by u, u resets and selects a random color. The probability for u to select v's color is $1/c\Delta$. Union bounding over all neighbors implies the sum in lemma. □

To complete the proof of the theorem, observe that the t_u values of neighbors of v are always higher than t, given that the neighbors started before v. This becomes obvious in Fig. 1 and establishes the following observation.

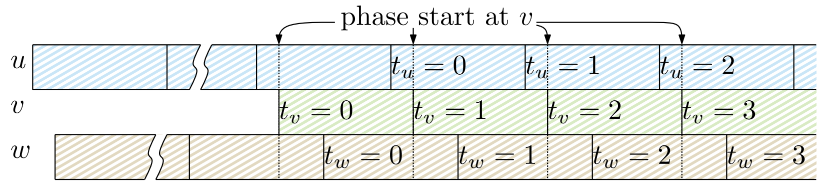


Figure 1: Illustration of the phases of v's neighbors u and w as seen during the analysis at v. Note that we abbreviate $t_u^{(v)}$ to t_u

Observation 5. *If v's neighbors started before v, it holds that $t = t_v \leq t_u$.*

Let us now prove the main result, which proves correctness of the algorithm. Note that the bound on the runtime of the algorithm at v is robust towards the wake-up of nodes anywhere in the network apart from the $\log n$-neighborhood of v. Although it is quite unlikely that v gets introduced to a conflict by a nearby node waking up, we must account for the probability here. For experimental evaluations regarding the impact of waking-up nodes, we refer to Section 4.

Theorem 6. *Let node v execute Algorithm 1. v computes a valid color w.h.p. $\mathcal{O}(\Delta \log n)$ time slots after (i) v started the algorithm and/or (ii) a node in v's $\log n$ neighborhood started the algorithm.*

Proof. In each round the algorithm transmits its current color and receives the colors of its neighbors with constant probability. Based on the probabilities for a successful transmission and the probability to select a color used by a neighbor after a conflict is detected, it holds that if the neighbors started before v (or at least before the analysis at v begins), we can prove that the probability decreases with each round. In order to decrease the probability with each round also for v's neighbors, and their neighbors, and so on, we require all nodes in a $\log n$ neighborhood of v to start before v, or postpone the analysis of v to a time t in which the nodes in the $\log n$ neighborhood started.

Let us consider rounds $0, \ldots, \log_k n$ such that in round 1 the nodes in the $\log n$ neighborhood of v are executing the algorithm. We prove the theorem by induction. In the first round, we apply the theorem to the nodes in N_v^{j-1}, then to N_v^{j-2}, and so on - until after l rounds the theorem is only applied to v, with the result that the conflict probability of v is at most $\frac{1}{k^{\log_k n}} = 1/n$.

Let us now prove the claim, and let therefore $j := \log_k n - t$. Our induction hypothesis is, that in round $t \in [\log_k n]$ it holds $\forall u \in N_v^{\log_k n - t} = N_v^j$:

1) $\Pr(\mathrm{cfl}^{t_u}(u)) \leq 1/k^t$

2) $t \leq t_u$ (see Observation 5),

This implies that we can apply Theorem 2 to the nodes in N_v^{j-1}. As a base case, we consider round $t = 0$. In this round it holds that all nodes in the $\log_k n$ neighborhood execute the algorithm, thus $\Pr(\mathrm{cfl}^{t_u}(u)) \leq 1$, and $0 \leq t_u$. Let us now assume the induction hypothesis is true for all $t' \leq t \in [\log_k n]$, and prove that it is also true for $t + 1$. It holds that for each node u in $N_v^j = N_v^{\log_k n - t}$ that the failure probability $\Pr(\mathrm{cfl}^{t_u}(u))$ is at most $1/k^t$, and that

$t \leq t_u$. The second part of the hypothesis holds trivially, as $t+1 \leq t_u+1$ follows directly from $t \leq t_u$. To show the first part of the hypothesis is true for $t+1$, we apply Theorem 2 to the nodes in N_v^j. This yields that $\Pr(\mathrm{cfl}^{t_u+1}(u)) \leq 1/k^{t+1}$ for $u \in N_v^{j-1} = N_v^{\log_k n-(t+1)}$. Thus, the hypothesis holds for $t+1$. $\square$

We have shown in this section that Algorithm 1 computes a valid $c\Delta$ coloring in $\mathcal{O}(\Delta \log n)$ time slots, for a constant $c > 3$. Since it is unclear how colors can be finalized, there is some probability that a conflict is introduced to a node if a nearby node wakes up. Thus, our bound holds without restriction only after all nodes in the $\log n$ neighborhood of a node are awake. We consider the practical implications of this restrictions in Section 4.4 of our experimental evaluation of the algorithm.

4 Experimental Evaluation

Let us now consider the practicability of our node coloring algorithm RandCDeltaColor (Algorithm 1). The algorithm is very simple, which allows an easy and straight-forward implementation in the used network simulator Sinalgo [11]. We shall show in this section that Rand4DeltaColor ($c = 4$) validly colors a network with 4Δ colors in time less than required for each node in the network to transmit one (the same) message to its neighbors, i.e., our coloring scheme is faster than local broadcasting. Note that despite local broadcasting can be seen as a lower bound for distributed node coloring on the theoretical side, this result is reasonable from a practical perspective.

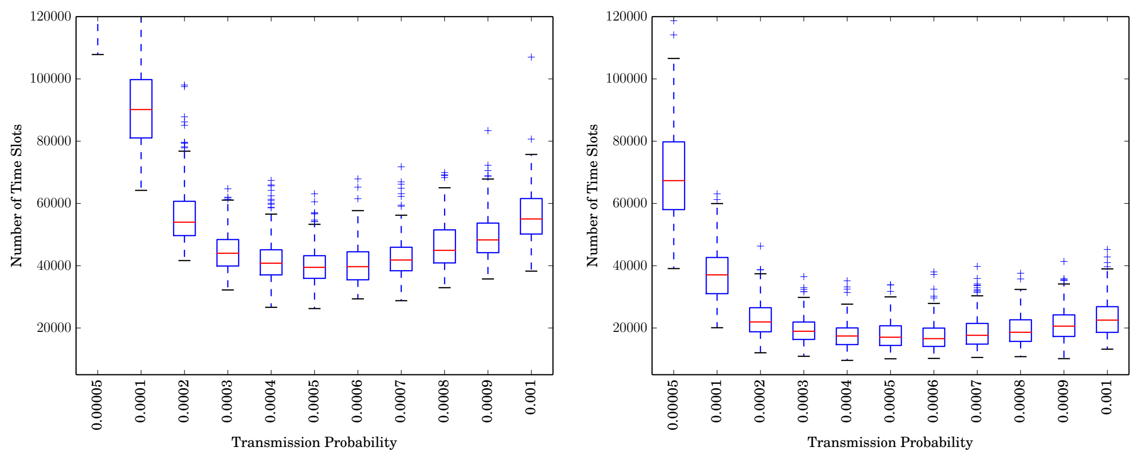
4.1 Experimental Setup

Our experiments are conducted with Sinalgo [11], an open-source simulation framework for networks algorithms in Java. It has build-in support for a variety of communication and interference models. We use wireless communication with SINR model of interference as describe in Section 2 with an attenuation coefficient of $\alpha = 6$, a threshold of $\beta = 1$. For simplicity we do not consider environmental noise (i.e., noise is set to 0). Our communication graph can be seen as a unit disk graph, due to the uniform transmission power of $P = 1$. We use a deployment area of 1000 by 1000, the nodes are deployed randomly of the area, and the broadcasting range is set to 100. We do not allow mobility of the nodes, simultaneous reception of multiple packets of receiving packets while transmitting a packet. We use the asynchronous simulation, which does not assume global rounds but is solely based on events, however, our nodes have a local clock which allows them to determine (for example) when one phase of Algorithm 1 is over. The most basic unit, the time required to transmit one message is considered to be 1, which we consider as one time slot. We set the length of each phase to 50 time slots. To implement asynchronous wake-up of our nodes, the nodes wake up within the first 50 time slots for both algorithms. The measured runtime begins with the globally first time slot. Our nodes do not know the maximum degree Δ in the network, but use a local estimate, i.e., their number of neighbors to determine the number of available colors. As this is less than Δ, the results reported in the following are expected to be slightly worse compared to those using Δ—but also more practical as Δ might not be known in practice.

For local broadcasting we measure the number of time slots required for all nodes in the network to finish, while for our coloring algorithm we measure the time until all nodes have a valid color, i.e. until there is no conflict in the network. All experiments are conducted using 200 runs with differing random seeds (randomization is used for deployment and determining whether to transmit in a time slot or not, depending on the transmission probability). We usually report median values. Our standard boxplots show the range of the values (apart from outliers) by a dashed interval, with the first and the third quartile within the box, and the median marked by a red line.

4.2 Optimal Transmission Probabilities

As our communication is based on probabilistic medium access, we must determine optimal transmission probabilities for the algorithm. The optimal transmission probabilities varies depending on the number of nodes deployed on the area. To determine optimal transmission probabilities for each setting, we execute both local broadcasting and Rand4DeltaColoring using a range of transmission probabilities. The results for a de-



(a) Local Broadcasting (b) Rand4DeltaColoring

Figure 2: Runtime of the algorithms for various transmission probabilities using 1000 nodes.

ployment of 1000 nodes are depicted in Fig. 2. We observe that the optimal transmission probability is comparable for both algorithms, and in the range of $[0.0005, 0.0006]$. Additionally, we can see that Rand4DeltaColor is a little more robust regarding a higher transmission probability, as the runtime increases slightly faster when deviating from the optimal transmission probability in the case of local broadcasting.

Using this method we can obtain optimal transmission probabilities along with the median number of required time slots to finish the respective algorithm. We report the optimal transmission probabilities (regarding the median number of time slots required) for local broadcasting and our coloring algorithm in Table 1. Note that we set our transmission probabilities uniformly for all nodes in the network, regardless of their degree, as this yielded a preferable runtime in our setting for both local broadcasting and Rand4DeltaColor. It would be interesting to see whether this also holds for a broader set of experiments.

Table 1: Transmission probabilities for a varying number of deployed nodes.

Number of Nodes		1000	2000	3000	4000	5000
Local Broadcasting	Transm. Prob.	0.0005	0.0002	0.0002	0.0001	0.00009
	Req. slots	39,485	88,185	140,268	192,887	242,833
Rand4DeltaColor	Transm. Prob.	0.0006	0.0003	0.0002	0.0001	0.0001
	Req. slots	16,575	39,073	62,319	88,039	109,269

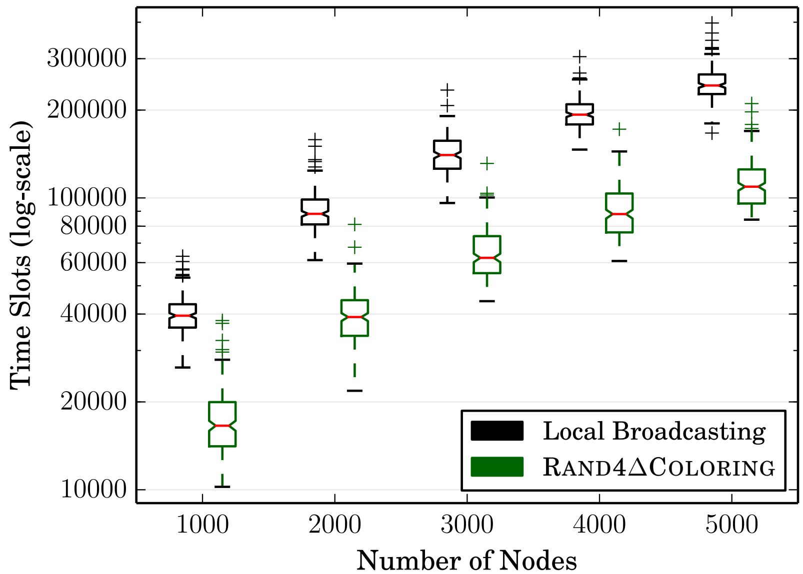


Figure 3: Runtime of the local broadcasting and the Rand4DeltaColor algorithm for a deployment between 1000 and 5000 nodes on our 1000×1000 area.

4.3 Comparing Rand4Delta with Local Broadcasting

Let us now directly compare the median running times of Rand4DeltaColor and local broadcasting. We use the optimal transmission probabilities determined in the previous section and compare the runtime of the algorithms for an increasing number of nodes. We consider a deployment between 1000 and 5000 nodes on the 1000×1000 area. It can be obtained from Fig. 3, that not only the median values (cf. Table 1) of the number of time slots required to finish our coloring algorithm is considerable lower than those for local broadcasting, but that this holds for the vast majority of the runs. Regardless of the number of deployed nodes, the values do hardly (if at all) intersect, apart from few outliers. Thus, we conclude that the proposed coloring algorithm is extremely fast, requiring even less time to finish than one round of local broadcasting in our setting. We expect this result to be robust against a variation of the parameters such as the number of nodes, the area or the SINR constants, although actual runtime values may vary.

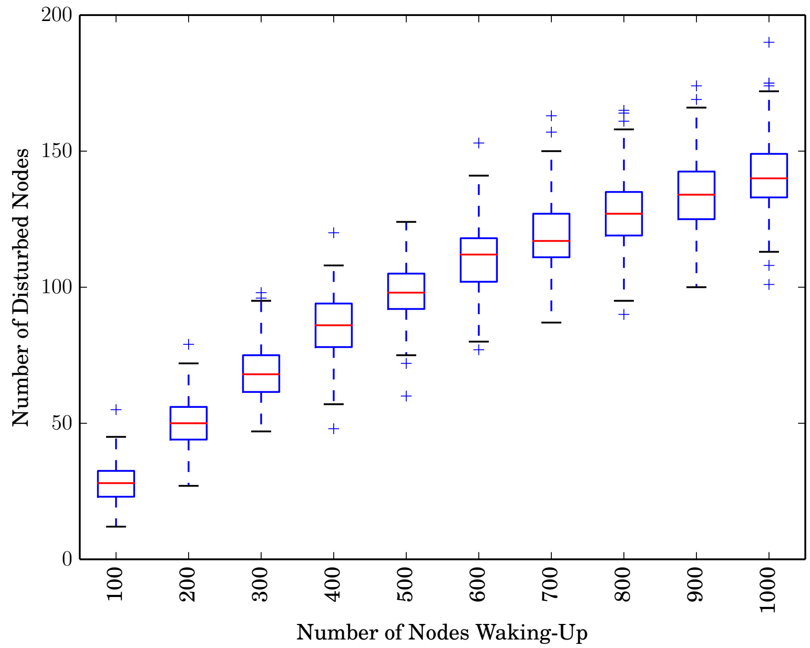


Figure 4: Assume a network of 1000 colored nodes. Let a varying number of nodes wake-up and count the number of already colored nodes that are disturbed.

4.4 The Impact of Nodes Waking-Up Late

We concluded that our coloring algorithm is very fast in general. However, in our bound on the runtime we could not guarantee that nodes close to other nodes that wake-up can maintain their color. Specifically, nodes in the $\log n$ neighborhood of a newly awaking node may be introduced to a conflict with a small but non-negligible probability. We shall consider the influence of nodes waking up after other nodes have colored in this section. Therefore consider the following setting: A set of 1000 nodes are awake and execute Rand4DeltaColoring until they are colored. Afterwards, an additional set of nodes awake and start executing the Rand4DeltaColoring. Our measure for this experiment is the number of nodes of the first set that are *disturbed*, which means that one of their neighbors selected its color. We depict the results in Fig. 4. Note that the "disturbed" nodes, are not necessarily disturbed in the sense that they got to know of the conflict or even changed their color. Thus, although we might over-estimate the damage introduced by waking-up nodes, each node that wakes up in the network disturbed approximately 0.15 to 0.25 nodes on average, while the percentage decreases for an increasing number of nodes that wake-up after the 1000 already colored nodes.

Also, observe that our algorithm in its current stage does not implement any methods to respect the colors of already colored nodes. We expect this value to be even less if a listen phase would be added to the algorithm and used colors would be respected. However, such a modification might require an increased number of available colors (i.e., $c = 5$) in the algorithm to guarantee correctness. For practical applications, we have seen that (without such a modification) even a decrease to $c = 1$ works well in practice, finishing

5 Conclussion

We have shown in this work that the analysis of the simple coloring algorithm RandCDeltaColor by Fuchs and Prutkin [1] can be generalized to the asynchronous setting without modification of the algorithm itself. Additionally, we observed in our experimental evaluation that the algorithm is very fast. Using $c = 4$, Rand4DeltaColor computes a valid 4Δ coloring of the network in less time than required to finish a local broadcast for each node in the network.

Regarding future work, we are interested in making the algorithm more robust towards the late wake-up of nodes by respecting the colors already selected by neighboring nodes.

References

[1] F. Fuchs and R. Prutkin, "Simple distributed $\Delta + 1$ coloring in the SINR model," 2014, under submission. [Online]. Available: http://arxiv.org/abs/1502.02426

[2] F. Fuchs and D. Wagner, "Local broadcasting with arbitrary transmission power in the SINR model," in *Proc. 21st Internat. Colloq. Structural Inform. and Communication Complexity (SIROCCO'14)*, ser. Lecture Notes Comput. Sci., M. M. Halldórsson, Ed., vol. 8576. Springer, 2014, pp. 180–193.

[3] M. M. Halldórsson and P. Mitra, "Towards Tight Bounds for Local Broadcasting," in *Proc. 8th ACM Internat. Workshop on Foundations of Mobile Computing (FOMC'12)*. ACM Press, July 2012.

[4] D. Yu, Q.-S. Hua, Y. Wang, and F. C. M. Lau, "An O(log n) Distributed Approximation Algorithm for Local Broadcasting in Unstructured Wireless Networks," in *Proc. 8th Internat. Conf. on Distributed Computing in Sensor Systems (DCOSS'12)*. IEEE Computer Society, 2012, pp. 132–139.

[5] D. Yu, Y. Wang, Q.-S. Hua, and F. C. M. Lau, "Distributed Local Broadcasting Algorithms in the Physical Interference Model," in *Proc. 7th Internat. Conf. on Distributed Computing in Sensor Systems (DCOSS'11)*. IEEE Computer Society, 2011, pp. 1–8.

[6] O. Goussevskaia, T. Moscibroda, and R. Wattenhofer, "Local Broadcasting in the Physical Interference Model," in *Proc. 5th ACM Internat. Workshop on Foundations of Mobile Computing (DialM-POMC'08)*. ACM Press, 2008, pp. 35–44.

[7] D. Yu, Y. Wang, Q.-S. Hua, and F. C. M. Lau, "Distributed $(\Delta + 1)$ Coloring in the Physical Model," in *Proc. 7th Internat. Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS'11)*, ser. Lecture Notes Comput. Sci., T. Erlebach, S. E. Nikoletseas, and P. Orponen, Eds., vol. 7111. Springer, 2011, pp. 145–160.

[8] B. Derbel and E.-G. Talbi, "Distributed Node Coloring in the SINR Model," in *Proc. 30th Internat. Conf. on Distributed Computing Systems (ICDCS'10)*. IEEE Computer Society, 2010, pp. 708–717.

[9] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *SIAM J. Comput.*, vol. 15, no. 4, pp. 1036–1053, 1986.

[10] I. Finocchi, A. Panconesi, and R. Silvestri, "An experimental analysis of simple, distributed vertex coloring algorithms," *Algorithmica*, vol. 41, no. 1, pp. 1–23, 2005.

[11] Distributed Computing Group, ETH Zurich, "Sinalgo - simulator for network algorithms," 2008, version 0.75.3. [Online]. Available: http://sourceforge.net/projects/sinalgo/