Simple Distributed $\varDelta + 1$ Coloring in the SINR Model*

Fabian Fuchs and Roman Prutkin

Karlsruhe Institute for Technology (KIT)
Karlsruhe, Germany
{fabian.fuchs,roman.prutkin}@kit.edu

Abstract. In wireless ad hoc networks, distributed node coloring is a fundamental problem closely related to establishing efficient communication through TDMA schedules. For networks with maximum degree $\varDelta$, a $\varDelta + 1$ coloring is the ultimate goal in the distributed setting as this is always possible. In this work we propose a very simple $4\varDelta$ coloring along with a color reduction technique to achieve $\varDelta + 1$ colors. All algorithms have a runtime of $\mathcal{O}(\varDelta \log n)$ time slots. This improves on previous algorithms for the SINR model either in terms of the number of required colors or the runtime, and matches the runtime of local broadcasting in the SINR model (which can be seen as an asymptotical lower bound).

1 Introduction

One of the most fundamental problems in wireless ad hoc or sensor networks is efficient communication. Indeed, most algorithms concerned with the physical or *Signal-to-Interference-and-Noise-Ratio* (SINR) model consider algorithms to establish initial communication right after the network begins to operate. However, those initial methods of communication are not very efficient, as there are either frequent collisions and reception failures due to interference, or time is wasted in order to provably avoid such collisions and failures. If local broadcasting [10, 13, 18] is used, a multiplicative $\mathcal{O}(\varDelta \log n)$ factor is required to execute message-passing algorithms in the SINR model, where $\varDelta$ is the maximum degree in the network (we use a broadcasting range to define neighborhood in the SINR model, cf. Section 2). Thus, wireless networks often use a more refined transmission schedule as part of the Medium Access Control (MAC) layer. One of the most popular solutions to the medium access problem are Time-Division-Multiple-Access (TDMA) schedules, which provide efficient communication by assigning nodes to time slots. The main problem in establishing a TDMA schedule can be reduced to a distributed node coloring. Given a node coloring, we can establish a transmission schedule by simply associating each color with one time slot. The node coloring considered in this work ensures that two nodes capable of communicating directly with each other do not select the same color. Note that a TDMA schedule based on such a coloring is not yet feasible in the SINR model.

* The full version of this work is available as [6].

C. Scheideler (Ed.): SIROCCO 2015, LNCS 9439, pp. 149–163, 2015.
DOI: 10.1007/978-3-319-25258-2_11

However, a feasible TDMA schedule can be computed based on our coloring, for example as shown in [3,7].

The problem of distributed node coloring dates back to the early days of distributed computing in the mid-1980s. In contrast to the centralized setting, a $\Delta + 1$ coloring is considered to be the ultimate goal in distributed node coloring as it is already NP-complete to compute the chromatic number (i.e., the minimum number of colors required to color the graph) in the centralized setting [9]. There is a rich line of research in this area, however, most of the work has been done for message-passing models like the $\mathcal{LOCAL}$ model. Such models are designed for wired networks and do not fit the specifics of wireless networks.

In the SINR model, also denoted as the physical model due to its common use in electrical engineering, wireless communication is modelled based on the signal transmission and a geometric decay of the signal strength. It improves on other models for wireless communication, such as the protocol model, which considers interference as a local and binary property by declaring a transmission to be successful iff it is not in the interference range of another transmitting node. It has been shown that such models are quite limited, as protocols designed for the SINR model surpass the theoretically achievable performance of protocols designed for the protocol model [15]. In this work, we use two simple and well-known algorithms (covered for example in [2]) designed for message-passing models, and show that we can *efficiently* execute the algorithms in the SINR model. However, this cannot be achieved by a simple simulation of each round of the message passing algorithm by one execution of local broadcasting as this results in a runtime of $\mathcal{O}(\Delta \log^2 n)$ time slots. Instead, we modify both the communication rounds in the SINR model and the algorithms to perfectly fit together. The synergy effect of our careful adjustments is that the coloring algorithm runs in $\mathcal{O}(\Delta \log n)$ time slots, which is asymptotically exactly the runtime of one local broadcast [10]. This matches the runtime of current $\mathcal{O}(\Delta)$ coloring algorithms [3], and improves on current $\Delta + 1$ coloring algorithms for the SINR model which require $\mathcal{O}(\Delta \log n + \log^2 n)$ or $\mathcal{O}(\Delta \log^2 n)$ time slots [19].

The communication between nodes in our algorithm is based on the local broadcasting algorithm proposed by Goussevskaia *et al.* [10]. Thus, we require the nodes to know an upper bound on the maximum number of nodes in a node's surroundings (which we call proximity area, cf. Section 2), an upper bound on the number of nodes in the network, as well as some model-related hardware constants in order to enable initial communication. All our results hold *with high probability* (w.h.p.), i.e., with probability at least $1 - \frac{1}{n^c}$, where n is the number of nodes, and $c \geq 1$ a constant. As union bounding a w.h.p. event only decreases the constant c, resulting in a constant increase in the runtime, we refrain from stating exact w.h.p. bounds in our analysis to simplify notation. Note that such requirements and assumptions are common in the SINR model.

1.1 Related Work and Contributions

Due to the rich amount of work on distributed node coloring in the message-passing model, we refer to a recent monograph by Barenboim and Elkin [2] for a thorough overview on distributed graph coloring. In wireless networks, the SINR model received increasing attention first in the electrical engineering community, and was picked up by the algorithms community due to a seminal work by Gupta and Kumar [12]. An overview of works regarding transmission scheduling in the SINR model can be found in a survey by Goussevskaia, Pignolet and Wattenhofer [11]. A coloring algorithm due to Moscibroda and Wattenhofer [14] has been adapted to the SINR model by Derbel and Talbi [3], and extended to support directed communication by Fuchs and Wagner [8]. Derbel and Talbi provide an algorithm that computes an $\mathcal{O}(\Delta)$ coloring in $\mathcal{O}(\Delta \log n)$ time slots. Their algorithm first computes a set of leaders using a maximal independent set (MIS, cf. Section 2) algorithm, then leader nodes assign colors to non-leaders, which again compete for their final color with a restricted number of neighboring nodes that may have received the same assignment. Yu *et al.* [19] propose two $\Delta + 1$ coloring algorithms that do not require the knowledge of the maximum node degree Δ. Their first algorithm runs in $\mathcal{O}(\Delta \log n + \log^2 n)$ time slots and assumes that nodes are able to increase their transmission power for the computation. This prevents conflicts between non-leader nodes by allowing the set of leaders to directly communicate to other leaders outside the transmission region and thus coordinating the assignment process. Their second algorithm does not require this assumption, and runs in $\mathcal{O}(\Delta \log^2 n)$ time slots.

Our main contributions are 1. a simple and efficient 4Δ coloring algorithm, requiring $\mathcal{O}(\Delta \log n)$ time slots; 2. an abstract method that has the potential of improving the runtime of other randomized algorithms in the SINR model by a $\log n$ factor; and 3. an asynchronous color reduction scheme, which, combined with known coloring algorithms computes a $\Delta + 1$ coloring in overall $\mathcal{O}(\Delta \log n)$ time slots. Also, the color reduction simplifies to an almost trivial color reduction scheme yielding the same results restricted to the synchronous setting.

The coloring algorithms improve current algorithms in the same setting (cf. Derbel and Talbi [3]) regarding the number of colors, and achieve the declared goal of $\Delta+1$ colors, while the runtime is matched. Other $\Delta+1$ coloring algorithms in the SINR model require at least $\mathcal{O}(\Delta \log n + \log^2 n)$ time slots (under non-comparable assumptions). Our new method to improve the runtime by a $\log n$ factor carefully combines the uncertainty in randomized algorithms with the uncertainty in the SINR model to handle them simultaneously in the analysis. For more details, we refer to the Analysis of Algorithm 1 in Section 3.1.

Roadmap: In the next section we state the model along with required definitions. In Section 3 the simple 4Δ coloring algorithm is described and analyzed. We introduce and analyse the color reduction scheme in Section 4.

2 Model and Preliminaries

The *Signal-to-Interference-and-Noise-Ratio* (SINR) model is used to model if a transmission in a wireless network can be successfully decoded at the intended receivers or not. We say that a transmission from a sender to a receiver is *feasible* if it can be decoded by the receiver. In the SINR model it depends on the ratio between the desired signal and the sum of interference from other nodes plus the background noise whether a certain transmission is successful. Let each node v in the network use the same transmission power P. Then a transmission from u to v is feasible if and only if $\frac{\frac{P}{\mathrm{dist}(u,v)^{\alpha}}}{\sum_{w \in I} \frac{P}{\mathrm{dist}(w,v)^{\alpha}} + \mathrm{N}} \geq \beta$, where α, β are constants depending on the hardware, N reflects the environmental noise, $\mathrm{dist}(u, v)$ the Euclidean distance between two nodes u and v, and $I \subseteq V$ is the set of nodes transmitting simultaneously to u. The *broadcasting range* r_B of a node v defines the range around v up to which v's messages should be received. We denote the set of *neighbors* of v by $N_v := \{w \in V \backslash \{v\} \,|\, \mathrm{dist}(v, w) \leq r_B\}$ and $N_v^+ := N_v \cup \{v\}$. Based on the SINR constraint, the *transmission range* of $r_T \leq (\frac{P}{\beta \mathrm{N}})^{1/\alpha}$ is an upper bound for the broadcasting range (with $r_B < r_T$ to allow multiple simultaneous transmissions). Let the *broadcasting region* B_v be the disk with range r_B centered at v. To prove successful communication within the broadcasting range, we need the concept of a *proximity range* $r_A > 2r_B$ around a node v as introduced in [10]. Let Δ_A^v be the number of nodes with distance less than r_A to v, and $\Delta^A := \max_{v \in V} \Delta_A^v$. It holds that $\Delta^A \in \mathcal{O}(\Delta)$. As further technical details of the proximity range are not required in our analysis, we refer to [10] or [6] for the exact definitions.

The communication graph $G = (V, E)$ is defined as follows. The set of vertices V in the graph corresponds to the set of nodes in the network, while there is an edge $(u, v) \in E$ if and only if u and v are neighbors (i.e., they are within each other's broadcasting range). The *maximum degree* in the network is $\Delta := \max_{v \in V} |N_v|$. Note that since $r_B < r_T$, a node v may successfully receive transmissions from nodes that are not its neighbors in the communication graph, although successful transmission from those nodes cannot be guaranteed. As the signal strength decreases geometrically in the SINR model, we assume that messages from outside the broadcasting range are discarded by considering the signal strength of a received message (usually provided by wireless receivers as the Received-Signal-Strength-Indication (RSSI) value [1]). Thus, the maximum degree is defined as for the $\Delta + 1$ coloring in [19]. In a more practical setting, one could also define the communication graph based on the actual communication between two nodes.

We call two nodes $v, u \in V$ *independent* if they are not neighbors. A set $S \subseteq V$ such that the nodes in S are pairwise independent is called *independent set.* Obviously, $S \subseteq V$ is a *maximal independent set* (MIS) if S is independent and there is no $v \in V \backslash S$ with $S \cup \{v\}$ independent. We denote the set of integers $\{0, \ldots, i\}$ by $[i]$. Let us now define the coloring problem. Given a set of nodes V so that each node $v \in V$ has a color c_v, and let d be an integer. Then V has a *valid* $d+1$ *coloring*, if for each node v holds $\forall w \in N_v : c_v \neq c_w$ and $c_v \in [d]$. Observe that in a valid coloring each color in the network forms a independent set.

In the *synchronous setting*, we assume that nodes start the algorithm at the same time. In the more realistic *asynchronous setting*, arbitrary wake-up of nodes is allowed, and we do not require synchronized time slots; precise clocks, however, are assumed. With the so-called ALOHA trick [16], e.g. as used in [10], we use time slots in our analysis, although the nodes do not assume common time slots.

The nodes use two different transmission probabilities in order to adapt to the requirements of the corresponding algorithms. Probability $p_1 := \frac{1}{2\Delta^A}$ is used in Algorithm 1, while Algorithm 2 uses p_1 and $p_2 := \frac{1}{180}$. If $p \geq c$ for probability p and a constant c, we say that p is at least constant, or simply constant. Let c be an arbitrary constant with $c > 1$. Throughout the paper, we use the following definitions: $\kappa_\ell := c\lambda \ln n / p_\ell$ for $\ell = 1, 2$, $\kappa_0 := \lambda \ln 12/p_1$, and λ a constant (for more details, we refer to [6]). Note that $\kappa_0 \in \mathcal{O}(\Delta)$, $\kappa_1 \in \mathcal{O}(\Delta \log n)$, and $\kappa_2 \in \mathcal{O}(\log n)$.

Extending Local Broadcasting: We show that local broadcasting with constant success probability in time reversely proportional to the transmission probability can be achieved. This extends known results regarding local broadcasting, which guarantee local broadcasting with *high* probability for a fixed number of time slots. Although we are the first to use local broadcasting with constant success probability, the proof of the following lemma is mainly based on standard techniques. Thus, we defer it to [6] due to space constraints.

Lemma 1. *Let v be a node transmitting with probability p_1, then it successfully transmits to its neighbors with probability $\geq 11/12$ within κ_0 time slots. Transmissions with probability p_ℓ for κ_ℓ time slots are successful w.h.p. for $\ell \in \{1, 2\}$.*

3 Simple 4Δ Coloring

The algorithm we propose is at its heart a very simple and well-known randomized coloring algorithm. The underlying approach is well-known, and for example covered in [2, Chapter 10]. Essentially, this kind of algorithms draw a random color whenever two neighboring nodes have the same color (i.e., there is a conflict between them). Our first algorithm, Rand4DeltaColoring (Algorithm 1), is a simple, phase-based coloring algorithm. We say that two neighbors v, w have a *conflict* if $c_v = c_w$ and denote the temporary color of v in phase t by c_v^t. In each phase t the node v checks whether it knows of a conflict with one of its neighbors. The set of neighbors that are in a conflict with v in phase t is $X^t(v) := \{w \in N_v | c_v^t = c_w^t\}$. We call $X^t(v)$ the *conflict set* of v in phase t, and denote the event that v is in a conflict in phase t by $\mathcal{E}_{\text{confl}}^t(v) := \exists w \in X^t(v)$. Note that there may be nodes in $X^t(v)$, for which v is not aware of the conflict (due to the uncertainty in the nodes communication), however, this does not affect the event. If a conflict is detected by v, the node randomly draws a new

Algorithm 1. Rand4DeltaColoring for node v

```
1 $F_v \leftarrow [4\Delta]$, $c_v^{-1} \leftarrow F_v$.rand()
2 for $t \leftarrow 0; t \leq 6(c+3)\ln n; t \leftarrow t+1$ do                    // each phase
3     if $c_v^{t-1} \notin F_v$ then $c_v^t \leftarrow F_v$.rand()             // if conflict, new color
4     else $c_v^t \leftarrow c_v^{t-1}$                                       // otherwise, keep it
5     $F_v \leftarrow [4\Delta]$
6     Transmit $c_v^t$ with probability $p_1$ for $\kappa_0$ time slots
7     foreach received color $c_w^t$ from neighbor $w \in N_v$ do $F_v \leftarrow F_v \setminus \{c_w^t\}$
```

color from the set F_v of colors not taken by a neighbor in the previous phase and transmits this color in the current phase. The event that a transmission from v to all neighbors N_v of v in phase t is *successful* is $\mathcal{E}^t_{\text{succ}}(v)$. A transmission from v to its neighbors in phase t is not successful or *fails* if at least one neighbor was unable to receive the message. The corresponding event is $\mathcal{E}^t_{\text{fail}}(v)$. We replace $\mathcal{E}$ by $\mathbb{P}$ to denote the probability of an event, e.g. $\mathbb{P}^t_{\text{succ}}(v)$ for $\mathcal{E}^t_{\text{succ}}(v)$. Note that although the events $\mathcal{E}^t_{\text{succ}}(v)$, and $\mathcal{E}^t_{\text{fail}}(v)$ may not be independent of events happening at other nodes, our bounds on the corresponding probabilities $\mathbb{P}^t_{\text{succ}}(v)$ and $\mathbb{P}^t_{\text{fail}}(v)$ are independent from the node v and possible events at other nodes. Also, our bounds $\mathbb{P}^t_{\text{succ}}(v)$ and $\mathbb{P}^t_{\text{fail}}(v)$ on these events include the event that v reaches some but not all of its neighbors, as $\mathbb{P}^t_{\text{fail}}(v) \leq 1 - \mathbb{P}^t_{\text{succ}}(v) \leq 1/12$ and $11/12 \leq \mathbb{P}^t_{\text{succ}}(v) \leq 1$ (see Lemma 1). Finally, the phase is concluded by transmitting the current color. This computes a valid coloring with 4Δ colors in $\mathcal{O}(\log n)$ phases, while each phase takes $\mathcal{O}(\Delta)$ time slots. In contrast to previous algorithms of this kind, we do not assume that successful communication is guaranteed by lower layers. Instead we allow the uncertainty in the randomized algorithm to be combined with the uncertainty in the communication in the SINR model, which is jointly handled in the analysis. Thereby we can reduce the number of time slots required for each phase by a $\log n$ factor (from $\mathcal{O}(\Delta \log n)$ for the trivial analysis to $\mathcal{O}(\Delta)$), making this simple approach viable in the SINR model. Thus, Algorithm 1 solves the node coloring problem using 4Δ colors in $\mathcal{O}(\Delta \log n)$ time slots, which matches the runtime of local broadcasting in the SINR model and improves the state-of-the-art $\mathcal{O}(\Delta)$ coloring in [3]. Let us now state the main results of this section.

Theorem 2. *Let all nodes start executing Algorithm 1 simultaneously. After the execution, all nodes have a valid color $c_v \leq 4\Delta$ w.h.p.*

For the asynchronous setting, the bound on the runtime holds for node v only after all nodes in v's $\log n$ neighborhood are awake

Corollary 3. *Let a node v execute Algorithm 1 in the asynchronous setting. Then v has a valid color $c_v \leq 4\Delta$ w.h.p., at most $\mathcal{O}(\Delta \log n)$ time slots after all nodes in its $\mathcal{O}(\log n)$-neighborhood started executing the algorithm.*

In the following section we prove the result for the synchronous setting. In Section 3.2 we briefly discuss extending it to the asynchronous setting. Our experiments in Section 3.3 show that the algorithm is very fast and robust even in the asynchronous setting.

3.1 Analysis of Rand4DeltaColoring

Despite the fact that the underlying coloring algorithm is well-known, our analysis is new and quite involved. The main reason for this is the uncertainty in whether a message is successfully delivered in one phase of Algorithm 1. In contrast to guaranteed message delivery, based for example on local broadcasting, message delivery with constant probability can be achieved a logarithmic factor faster, see Lemma 1. However, this reduction in runtime comes at a cost: While in the guaranteed message delivery setting, a node v can finalize its color once a phase without a conflict at v happened, this is not possible in our setting. We cannot guarantee the validity of the colors even if a node did not receive a message implying a conflict in one phase, as message transmission is successful only with constant probability. Nevertheless, we can show that after $\mathcal{O}(\log n)$ phases of transmitting the selected color and resolving eventual conflicts, the coloring is valid in the entire network w.h.p.

In order to prove correctness of Algorithm 1 (Rand4DeltaColoring) we shall first bound the probability of a conflict propagating from one phase of the algorithm to the next. This is the foundation for the result that our algorithm computes a valid 4Δ coloring in $\mathcal{O}(\Delta \log n)$ time slots w.h.p. for both the synchronous and the asynchronous setting. Assuming that a node v has a conflict in phase t, there are only two cases that may lead to a conflict at v in phase $t+1$:

1. Node v had a conflict in phase t, and it did not get resolved (either due to being unaware of the conflict or since the new color implies a conflict as well).
2. A neighbor of v had a conflict in phase t and introduced the conflict by randomly selecting v's color.

We shall show that the probability for both cases is at most constant (see Lemma 4). Thus, after $\mathcal{O}(\log n)$ phases it holds with high probability that a valid color has been found. Note that the results in this section are restricted to the synchronous setting, however, they can be extended to the asynchronous case, cf. Section 3.2.

Lemma 4. *Let v be an arbitrary node and $\mathbb{P}^t_{\text{confl}}(v)$ the probability of a conflict at v in phase t. Then the probability of a conflict at v in phase $t+1$ is at most*

$$\mathbb{P}^{t+1}_{\text{confl}}(v) \leq \frac{5}{6} \cdot \max_{w \in N_v} \mathbb{P}^t_{\text{confl}}(w).$$

Proof. We shall prove the lemma by considering the two cases that may lead to a conflict at node v in phase $t+1$. The **first** case is that v has a conflict with at least one of its neighbors. Depending on which transmissions are successful there are 3 subcases. Note that $\rightarrow$ denotes $\mathcal{E}^t_{\text{succ}}(v)$, while $\leftarrow$ denotes $\exists w \in X^t(v) : \mathcal{E}^t_{\text{succ}}(w)$—with negations accordingly[1].

(a) $\not\rightarrow, \not\leftarrow$: It is not guaranteed that any of the conflict partners know of the conflict, as the transmissions from v and the nodes in the conflict set $X^t(v) \neq \emptyset$ failed at least partially. There is at least one neighbor $u \in X^t(v)$ that failed to transmit its color successfully to v, which happens with probability $\mathbb{P}^t_{\text{fail}}(u)$. Combined with v's failure to transmit its color successfully, case 1(a) happens

[1] A partial success of transmission is often sufficient to trigger dealing with a conflict. We do not consider this in our notation, however, as we evaluate $\mathbb{P}^t_{\text{succ}}(v)$ to be at most 1 for all v and since Pr(transmission from v to u fails) $\leq \mathbb{P}^t_{\text{fail}}(v) \leq 1/12$, our analysis covers this case.

with probability at most $\mathbb{P}^t_{\text{confl}}(v)(\mathbb{P}^t_{\text{fail}}(v)\Pr(\not\leftarrow)) \le \mathbb{P}^t_{\text{confl}}(v)\mathbb{P}^t_{\text{fail}}(v)\mathbb{P}^t_{\text{fail}}(u) \le \mathbb{P}^t_{\text{confl}}(v)(1/12)^2$. If any conflict partner knows of the conflict, the conflict would be resolved with a certain probability (as in the following cases). However, as this is not guaranteed, we account for the worst case: the conflict is not resolved and propagates to the next phase. Note that since this case happens only with a small probability, it holds that the total probability of case (a) and conflict at v in phase $t+1$ is small.

(b) $\rightarrow, \not\leftarrow$: All nodes in $X^t(v)$ failed to transmit successfully, but v transmitted successfully to all neighbors. Thus, all nodes in $X^t(v)$ know of the conflict, while v might be unaware of it. This case happens with probability at most $\mathbb{P}^t_{\text{confl}}(v) \cdot (\mathbb{P}^t_{\text{succ}}(v) \cdot \Pr(\not\leftarrow))$. The probability that a node $w \in X^t(v)$ selects v's color in phase $t+1$ is at most $\sum_{w \in X^t(v)} 1/|F_w|$ (even if v knows of a conflict and itself selects a new color). This results in an overall probability of at most

$$\begin{aligned}
&\mathbb{P}^t_{\text{confl}}(v) \cdot (\mathbb{P}^t_{\text{succ}}(v) \cdot \Pr(\not\leftarrow)) \cdot \sum_{w \in X^t(v)} \frac{1}{|F_w|} \\
&\le \mathbb{P}^t_{\text{confl}}(v) \left(\prod_{w \in X^t(v)} \mathbb{P}^t_{\text{fail}}(w) \right) \cdot \sum_{w \in X^t(v)} \frac{1}{|F_w|} \\
&\overset{x := |X^t(v)|}{\le} \mathbb{P}^t_{\text{confl}}(v) \left(\mathbb{P}^t_{\text{fail}}\right)^x \cdot \frac{x}{3\Delta} \le \frac{1}{3\Delta}\mathbb{P}^t_{\text{confl}} \cdot x \left(\frac{1}{12}\right)^x \le \frac{1}{24}\mathbb{P}^t_{\text{confl}}
\end{aligned}$$

where the first inequality holds since the event $\not\leftarrow$ is equivalent to $\forall w \in X^t(v) : \mathcal{E}^t_{\text{fail}}(w)$ and $\mathbb{P}^t_{\text{succ}}(v) \le 1$. The second inequality holds since $|F_w| \ge 3\Delta$ as w and v are uncolored and by setting $x = |X^t(v)|$. The last inequality holds since $x(1/12)^x \le 1/12$ for all $x \in \{1, \ldots, \Delta\}$, and $\Delta \ge 1$.

(c) $\leftarrow$: It holds that v knows of the conflict. Whether v's neighbors know of it or not is not guaranteed. This case happens with probability at most $\mathbb{P}^t_{\text{confl}}(v) \cdot (\Pr(\leftarrow))$. The probability that at least one neighbor of v has or selects the same color as v is at most $\sum_{w \in N_v} \frac{1}{|F_v|} \le |N_v| \frac{1}{3\Delta} \le \frac{1}{3}$.

Using $\Pr(\leftarrow) \le 1$, this results in a probability for a conflict at v in phase $t+1$ of at most $\mathbb{P}^t_{\text{confl}}(v) \cdot (1/144 + 1/24 + 1/3 \cdot \Pr(\leftarrow)) < \left(\frac{1}{2}\right) \cdot \mathbb{P}^t_{\text{confl}}$.

In the **second** case, there was no conflict at v in phase t, but a neighbor w of v selected v's color due to a conflict at w, which happens with probability at most

$$\begin{aligned}
&\sum_{w \in N_v} \underbrace{\Pr(c_v^{t+1} = c_w^{t+1})}_{v\text{'s neighbor } w \text{ selects } v\text{'s color}} \sum_{u \in N_w} \underbrace{\Pr(c_u^t = c_w^t)}_{\substack{u \in N(w) \text{ told } w \\ \text{about their conflict}}} \\
&\le \sum_{w \in N_v} \Pr(c_v^{t+1} = c_w^{t+1}) \mathbb{P}^t_{\text{confl}}(w) \\
&\le \sum_{w \in N_v} \frac{1}{|F_w|} \mathbb{P}^t_{\text{confl}}(w) \le \left(\frac{1}{3}\right) \max_{w \in N_v} \mathbb{P}^t_{\text{confl}}(w)
\end{aligned}$$

The last inequality holds since $\sum_{w \in N_v} \frac{1}{|F_w|} \leq \sum_{w \in N_v} \frac{1}{3\Delta} \leq \frac{1}{3}$. Combining all events that could lead to a conflict at v in phase $t+1$ it holds that the probability of the union of the events is at most

$$\mathbb{P}^{t+1}_{\text{confl}}(v) \leq \left(\frac{1}{2}\right) \mathbb{P}^{t}_{\text{confl}}(v) + \left(\frac{1}{3}\right) \max_{w \in N_v} \mathbb{P}^{t}_{\text{confl}}(w) \qquad \leq \frac{5}{6} \cdot \max_{w \in N_v^+} \mathbb{P}^{t}_{\text{confl}}(w),$$

which concludes the proof. □

Note that the second case could be avoided if message delivery in each phase would be guaranteed, as a node v that does not have a conflict in phase t, would simply finalize its current color and communicate this. Thus, v could not be forced into a conflict anymore. We shall now show that a set of nodes executing Algorithm 1 computes a valid coloring, and hence prove Theorem 2.

Proof (of Theorem 2). Let us consider the probability of a conflict at an arbitrary node $v \in V$ in phase $t = 6(c+3)\ln n$. It holds that

$$\begin{aligned}\mathbb{P}^{t}_{\text{confl}}(v) &\leq \left(\frac{5}{6}\right) \max_{w \in N_v} \mathbb{P}^{t-1}_{\text{confl}}(w) \leq \left(\frac{5}{6}\right) \max_{w \in V} \mathbb{P}^{t-1}_{\text{confl}}(w) \\ &\leq \left(\frac{5}{6}\right)^{t} \max_{w \in V} \mathbb{P}^{0}_{\text{confl}}(w) \leq \left(1 - \frac{1}{6}\right)^{6(c+3)\ln n} \leq \frac{1}{n^{c+3}},\end{aligned}$$

where the first inequality is due to Lemma 4. The third inequality holds since all nodes are in the same phase due to the synchronous start of the algorithm. Note that the upper bound on the probability that a conflict propagates holds for all nodes. The fourth inequality holds as $\mathbb{P}^{0}_{\text{confl}}(v) \leq 1$ for all nodes v. The last inequality holds due to a well-known mathematical fact (cf. [6]). Thus, the probability for a conflict at an arbitrary node v is small. A union bound over all nodes in the network implies that the coloring is valid w.h.p. The runtime of Algorithm 1 is $\mathcal{O}(\Delta \log n)$, as it consists of $6(c+3)\ln n = \mathcal{O}(\log n)$ phases, and each phase takes $\kappa_0 = \mathcal{O}(\Delta)$ time slots according to Lemma 1. □

3.2 Asynchronous Simple Coloring

Let us now briefly consider the asynchronous setting. For this section, we call all nodes that can reach v within $\mathcal{O}(\log n)$ rounds the neighborhood of v, and say that this neighborhood is stable if those nodes are all awake. If the neighborhood of a node v is stable, Lemma 4 holds as well, with only small changes to some constants in the proof [5]. Thus, once all nodes in v's neighborhood are awake, we can bound the probability using said lemma, and prove Corollary 3 analog to the proof of Theorem 2.

3.3 Experimental Evaluation

In our experiments, we evaluate RAND4DELTACOLORINGusing the well-known network simulator *sinalgo* [4]. We use between 500 and 2500 nodes, uniformly

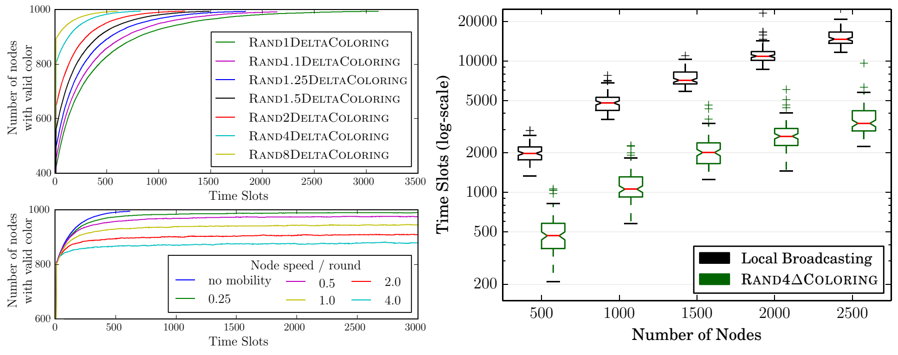

Fig. 1. Top left: Progress for varying number of available colors; Bottom left: Robustness under mobility constraints; Right: Runtime compared with local broadcasting

deployed on a square area of 1000×1000 meters. The SINR constants are set to $\alpha = 4$, $\beta = 10$, $\mathrm{N} = 1^{-9}$, $P = 1$, resulting in a transmission range of 100 meters. We set the broadcasting range to 84 meters, with average degree values ranging from 10 to about 50. We generally use asynchronous simulation and the nodes start at a random within the first 10 time slots. However, as sinalgo requires synchronous simulation for the mobility models, this experiment uses synchronized time slots. The time required to transmit one message is set to 1 time slot. We measure the number of time slots, and the number of nodes that have a valid color. The nodes do not know the global Δ value, but use the number of neighbors (plus one) as an estimate. More experiments are shown in the full version [6]. We observe three main points in Fig. 1. First, RAND4DELTACOLORING is very fast, requiring less time than one round of local broadcasting. Second, the algorithm is relatively robust, even under moderate mobility values of 1 meter per time slot, more than 90% of the nodes have a valid color (note that using mobile nodes, some color conflicts cannot be avoided). Finally, we can see that although our theoretical guarantees hold only for 4Δ colors, the algorithm can compute a valid coloring with only $\Delta + 1$ colors in our setting.

4 Asynchronous Color Reduction

In the following section we assume a valid node coloring with $d \in \mathcal{O}(\Delta)$ colors to be given and reduce the number of colors to $\Delta+1$ in $\mathcal{O}(d \log n)$ time slots. Let us first consider a very simple synchronous variant, which is also well known in the $\mathcal{LOCAL}$ model, cf. [2, Section 3.2]. In this variant, each node transmits its current color at the beginning of each round. Then, in the first round all nodes with color d select a color from the set $[\Delta]$, in the second round all those with color $d-1$, etc. This translates to an almost trivial (but new) color reduction scheme for the synchronous case, which we defer to [6] due to space constraints. The algorithm we present in this section circumvents the synchronization problem, essentially, by using two levels of MIS executions. Our algorithm is illustrated in Fig. 2,

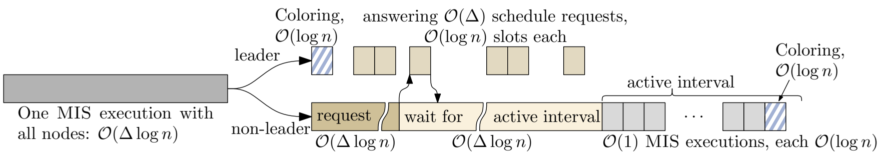
Coloring, $\mathcal{O}(\log n)$
answering $\mathcal{O}(\Delta)$ schedule requests, $\mathcal{O}(\log n)$ slots each
leader
Coloring, $\mathcal{O}(\log n)$
active interval
One MIS execution with all nodes: $\mathcal{O}(\Delta \log n)$
non-leader
request
wait for
active interval
$\mathcal{O}(\Delta \log n)$
$\mathcal{O}(\Delta \log n)$
$\mathcal{O}(1)$ MIS executions, each $\mathcal{O}(\log n)$

Fig. 2. Runtime. Overall $\mathcal{O}(\Delta \log n)$, given a $\mathcal{O}(\Delta)$ coloring.

the corresponding pseudocode can be found as Algorithms 2 to 5. We reference the MIS (Algorithm 3) executed with parameter $\ell = 1$ by *first level MIS*, and MIS($\ell = 2$) by *second level MIS*.

Let us now describe the algorithm in more detail. The algorithm starts by executing the first level MIS algorithm that determines a set of independent nodes, which we call leaders. Each leader node transitions to Algorithm 4, selects and transmits the color 0 it selected and initializes its periodic leader schedule. This schedule assigns each color an *active interval* of length $\mathcal{O}(\log n)$ time slots to allow the nodes of this color to select their final color from $[\Delta]$.

Each node v_i that is not in the first level MIS selects a leader from its broadcasting range and requests the relative time until it is v_i's turn to be *active*. Upon receipt of its active intervall, the node waits until the interval starts and then executes a second level MIS algorithm (which does not interfere with the first level MIS) for a constant number of times. In this second level MIS the algorithm benefits from fewer active nodes, and hence more efficient communication to allow each node to achieve successful transmission of a message to all neighbors in $\mathcal{O}(\log n)$ time slots. Moreover, we can speed up the MIS algorithm by the same factor of Δ to execute it in $\mathcal{O}(\log n)$ time slots, as only a constant number of nodes compete to be in each second level MIS. For each node that wins the second level MIS, there is no other node of the second level MIS in its broadcasting range. Thus, the winning node can select a valid color from $\{1, \dots, \Delta\}$ and transmit its choice to its neighbors without a conflict. If a node does not succeed to be in the second level MIS, it simply executes MIS(2) again. As each node succeeds in such an MIS within its active interval, each node selects one of the $\Delta + 1$ colors.

4.1 MIS, and Notation for AsyncColorReduction

Let us now describe the notation used in the algorithm. We denote the set of available colors by F_v. Note that throughout the algorithm, each node deletes the final colors it received from F_v. The MIS algorithm (Algorithm 3) aims at allowing exactly one node in each neighborhood to succeed to Algorithm Colored, select a color, and annouce its success in the MIS algorithm to its competitors. There are minor differences depending on the two levels $\ell = 1$ and $\ell = 2$, however, the algorithm remains the same. A description of the MIS algorithm can be found in [3], and the full version [6]. In Algorithm 4, v is a leader, col_v denotes the final color from $[\Delta]$, and Q is a queue used to store nodes w along with their initial color $\text{col}_w^{\text{tmp}}$ that request an active interval. The remaining time is based

Algorithm 2. AsyncColorReduction for node v

```
1  $F_v \leftarrow [\Delta]\setminus\{0\}$
2  foreach received $\mathrm{col}_w$ do continuously
3      $F_v \leftarrow F_v \setminus \{\mathrm{col}_w\}$
4  MIS(1)
```

Algorithm 4. Colored(ℓ) for node v

```
1  if $\ell = 1$ then    // Level 1 leader
2      $\mathrm{col}_v \leftarrow 0$, $Q \leftarrow \emptyset$, $c'_v = 0$
3      announce $M^1_C(v, \mathrm{col}_v)$ with prob. $p_2$ for $\kappa_2$ slots
4      Set $\tau(\mathrm{col}, c_v) \equiv \mathrm{col} \cdot \mu - c_v \bmod \Delta\mu$ neg., max., with $|\tau(\mathrm{col}, c_v)| > \kappa_2$
5      while protocol is executed do
6                                      // serve requests
7          $c'_v \leftarrow c'_v + 1$
8          transmit $M^1_C(v, \mathrm{col}_v)$ with probability $p_1$
9          foreach received request from neighbor w: $M_R(w, v, \mathrm{col}^{tmp}_w)$ do continuously
10             $Q$.push($(w, \mathrm{col}^{\mathrm{tmp}}_w)$)
11         if $Q$ not empty then
12             $(w, \mathrm{col}^{\mathrm{tmp}}_w) \leftarrow Q$.pop(), $t \leftarrow \tau(\mathrm{col}^{\mathrm{tmp}}_w, c'_v)$
13             for $\mathcal{O}(\log n)$ slots do
14                 transmit $M^1_C(v, w, t)$ with probability $p_2$
                   // inc. $c'_v$, $t$
15 else // Level 2 / Non-leader node
16     $\mathrm{col}_v \leftarrow F_v$.rand()        // valid
17     announce $M^2_C(c, \mathrm{col}_v)$ with prob. $p_2$ for $\kappa_2$ slots
18     while protocol is executed do
19                                     // keep color valid
20         transmit $\mathrm{col}_v$ with prob. $p_1$
```

Algorithm 3. MIS(ℓ) for node v, based on MW-coloring [3, 17]

```
1  $P_v = \emptyset$, $\textsc{Next} = \begin{cases} \textsc{Level2} & \text{if } \ell = 1 \\ \mathrm{MIS}(2) & \text{otherwise} \end{cases}$
2  for $\kappa_\ell$ time slots do       // Listen first
3      foreach $w \in P_v$ do $d_v(w) = d_v(w) + 1$
4      if $M^\ell_A(w, c_w)$ received then $P_v = P_v \cup \{w\}$; $d_v(w) = c_w$
5      if $M^\ell_C(w)$ received then Next(w)
6  $c_v = \Xi(P_v)$    // minimal, non-positive, not conflicting with competing counters in $P_v$
7  while true do // then compete for MIS
8      $c_v = c_v + 1$
9      if $c_v > \kappa_\ell$ then Colored($\ell$) // success
10     foreach $w \in P_v$ do $d_v(w) = d_v(w) + 1$
11     if $M^\ell_C(w)$ received then Next(w)
11     transmit $M^\ell_A(v, c_v)$ with probability $p_\ell$
12     if $M^\ell_A(w, c_w)$ received then
       // received competing counter
13         $P_v = P_v \cup \{w\}$; $d_v(w) = c_w$
14         if $|c_v - c_w| \leq \kappa_\ell$ then $c_v = \Xi(P_v)$
```

Algorithm 5. Level2(w) for node v with leader w

```
1  while true do
2      if $M^1_C(w, v, t)$ received then
3          while $t < 0$ do // wait for interval
4              $t \leftarrow t + 1$    // one time slot each
5          while $t < 2k^2\kappa_2$ do        // active interval
6              // increase $t$ by one in each time slot during MIS(2)
7              MIS(2)
8      else                 // transmit request
9          transmit $M_R(v, w, \mathrm{col}^{\mathrm{tmp}}_v)$ with probability $p_1$
```

on v's periodic schedule, which is defined by its counter value c'_v, and w's color. We set $k = 90$, which corresponds to the maximum number of active nodes in a broadcasting range, see Lemma 7. The function $\tau(\mathrm{col}_w, c_v)$ intuitively sets t to the start of the next interval corresponding to w's color in v's schedule, so that the starting time of w can be communicated by v w.h.p. before w's active interval starts. During the transmission interval, t is decreased appropriately.

Adapting the MIS Algorithm. We assume in the analysis that the MIS algorithm indeed computes a maximal independent set. Algorithm 3 is a simplification of the MIS part of the coloring algorithm in [3, 17], and therefore computes an MIS. Apart from constant changes, the lemma follows directly from Theorems 1 and 2 in [3] if $\ell = 1$, and from Lemma 1 along with setting Δ to a constant in the proofs of both theorems for $\ell = 2$.

Lemma 5. *Algorithm 3 computes an MIS among participating nodes in* $\delta_\ell \kappa_2 \in \mathcal{O}(\delta_\ell \log n)$ *time slots, where* $\delta_\ell = \begin{cases} \Delta & \text{if } \ell = 1 \\ k & \text{if } \ell = 2 \end{cases}$ *w.h.p.*

4.2 Analysis

Let us first state the main result of this section.

Theorem 6. *Given a valid node coloring with* $d \geq \Delta$ *colors, Algorithm 2 computes a valid* $\Delta + 1$ *coloring in* $\mathcal{O}(d \log n)$.

As the algorithm is essentially a simple color reduction scheme, each node selects a valid color if the communication can be realized as claimed. To prove this we show that in the second level indeed only a constant number of nodes are active in each broadcasting range (cf. Fig. 3). We use this to achieve message transmission from active nodes to all their neighbors in $\mathcal{O}(\log n)$ time slots, and show that the second level MIS can be executed in $\mathcal{O}(\log n)$ time slots. Finally, we prove that each non-leader node v succeeds in a second level MIS, and thus colors itself with a color from $[\Delta]$, within the active interval v is assigned by its leader. The proofs of the following lemmas are only given in the full version [6].

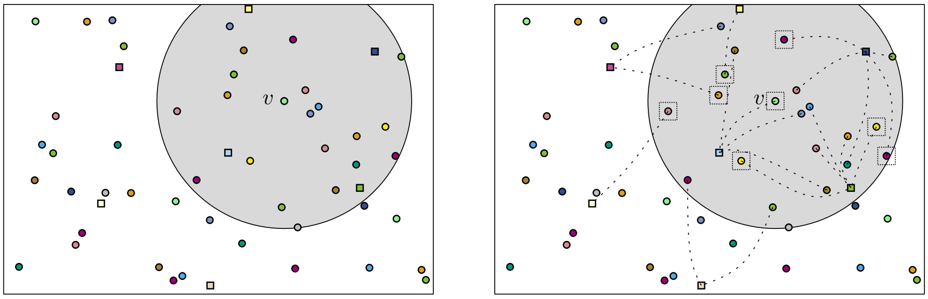

Fig. 3. Left: Node v with its broadcasting region in a network with valid coloring; Nodes in the first level MIS are squares. Right: Nodes in v's broadcasting range are connected to their selected leader by a dashed line. Nodes currently active in the second level are surrounded by a square.

Lemma 7. *In the second level, at most k nodes are active in each broadcasting range.*

The lemma follows from a geometric argument regarding the number of first level MIS nodes within a certain distance of each node. We use Lemma 7 to prove our bounds on the communication in Lemma 1. It allows us to increase the transmission probability in the second level MIS by a factor of Δ compared to classical local broadcasting, leading to a decrease in the time required for successful message transmission by the same factor of Δ to $\mathcal{O}(\log n)$. Based on this result we can bound the runtime of our algorithm, starting with Algorithm 5.

Lemma 8. *Let v execute Algorithm 5 with leader w. Then a) v transmits the request message successfully within κ_1 time slots w.h.p.; b) v receives its active interval after at most another κ_1 time slots w.h.p.; and c) the wait-time t until v's active interval starts is at most $\Delta 2k^2\kappa_2 \in \mathcal{O}(\Delta \log n)$.*

We shall now argue that each non-leader node succeeds to win a second level MIS in its active interval.

Lemma 9. *Given a node v executing Algorithm 5. Once $t = 0$, v wins a second level MIS set within $2k^2\kappa_2$ time slots.*

Essentially, this holds as there are multiple consecutive MIS executions, each allowing one node per broadcast range to win MIS, select a final color and withdraw. In the next MIS execution, another node wins, selects a color, etc. until all active nodes are colored. As a final step we show that the final color selected by each node is valid w.h.p.

Lemma 10. *Given a node v entering Algorithm 4. It holds that a) while v transmits its final color no neighbor of v succeeds in a second level MIS w.h.p.; and b) the color v selects is not selected by one of v's neighbors w.h.p.*

We are now able to prove the main theorem. Note that runtime bounds hold for each node once the node starts executing the algorithm.

Proof (Proof of Theorem 6). It follows from Lemma 10 and the fact that each node succeeds in an MIS (and hence enters Algorithm 4 and selects a final color), that the final color of each node is valid w.h.p. Only $\Delta + 1$ final colors are used, and a union bound over all nodes implies that the coloring is valid w.h.p. The first level MIS takes $\mathcal{O}(\Delta \log n)$ time slots according to Lemma 5. Algorithm 5 requires another $\mathcal{O}(\Delta \log n)$ slots until starting the active interval, which is of length $\mathcal{O}(\log n)$, resulting in $\mathcal{O}(\Delta \log n)$ time slots. □

Corollary 11. *Let each node in the asynchronous network execute the MW-coloring algorithm [3], followed by Algorithm 2. Then $\mathcal{O}(\Delta \log n)$ time slots after a node started executing the algorithms it selected a valid color from $[\Delta]$.*

5 Conclusion

We conclude that the proposed distributed 4Δ coloring algorithm is simple and very fast. RAND4DELTACOLORING performs well in our simulations, even in the asynchronous and mobile setting. Additionally, our color reduction scheme is the first $\Delta + 1$ coloring algorithm achieving a runtime of $\mathcal{O}(\Delta \log n)$, matching one round of local broadcasting.

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".

References

1. Bardwell, J.: Converting signal strength percentage to dbm values. WildPackets' White Paper (2002)
2. Barenboim, L., Elkin, M.: Distributed Graph Coloring: Fundamentals and Recent Developments. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers (2013)
3. Derbel, B., Talbi, E.G.: Distributed Node Coloring in the SINR Model. In: Proc. 30th Internat. Conf. on Distributed Computing Systems (ICDCS 2010). pp. 708–717. IEEE (2010)
4. Distributed Computing Group, ETH Zurich: Sinalgo - simulator for network algorithms (2008), `http://sourceforge.net/projects/sinalgo/`, version 0.75.3
5. Fuchs, F.: On asynchronous node coloring in the SINR model (2015), `http://i11www.iti.kit.edu/f-oancs-15.pdf` (unpublished manuscript)
6. Fuchs, F., Prutkin, R.: Simple distributed delta + 1 coloring in the SINR model. CoRR abs/1502.02426 (2015), `http://arxiv.org/abs/1502.02426`
7. Fuchs, F., Wagner, D.: On Local Broadcasting Schedules and CONGEST Algorithms in the SINR Model. In: Proc. 9th Internat. Workshop on Algorithmic Aspects of WSN (ALGOSENSORS 2013). pp. 170–184. Springer (2013)
8. Fuchs, F., Wagner, D.: Local broadcasting with arbitrary transmission power in the SINR model. In: Proc. 21st Internat. Colloq. Structural Inform. and Comm. Complexity (SIROCCO 2014), pp. 180–193. Springer (2014)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1979)
10. Goussevskaia, O., Moscibroda, T., Wattenhofer, R.: Local Broadcasting in the Physical Interference Model. In: Proc. 5th ACM Internat. Workshop on Foundations of Mobile Computing (DialM-POMC 2008), pp. 35–44. ACM (2008)
11. Goussevskaia, O., Pignolet, Y.A., Wattenhofer, R.: Efficiency of wireless networks: Approximation algorithms for the physical interference model. Foundations and Trends in Networking 4(3) (November 2010)
12. Gupta, P., Kumar, P.R.: The capacity of wireless networks. IEEE Trans. on Inform. Theory 46(2), 388–404 (2000)
13. Halldórsson, M.M., Mitra, P.: Towards Tight Bounds for Local Broadcasting. In: Proc. 8th ACM Internat. Workshop on Foundations of Mobile Computing (FOMC 2012). ACM (2012)
14. Moscibroda, T., Wattenhofer, M.: Coloring Unstructured Radio Networks. J. Distr. Comp. 21(4), 271–284 (2008)
15. Moscibroda, T., Wattenhofer, R., Weber, Y.: Protocol design beyond graph-based models. In: Proc. of the ACM Workshop on Hot Topics in Networks (HotNets-V), pp. 25–30 (2006)
16. Roberts, L.G.: Aloha packet system with and without slots and capture. SIGCOMM Comput. Commun. Rev. 5(2), 28–42 (1975)
17. Schneider, J., Wattenhofer, R.: Coloring unstructured wireless multi-hop networks. In: Proc. 28th ACM Symp. on Principles of Distributed Computing (PODC 2009), pp. 210–219. ACM (2009)
18. Yu, D., Hua, Q.S., Wang, Y., Lau, F.C.M.: 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 2012), pp. 132–139. IEEE (2012)
19. Yu, D., Wang, Y., Hua, Q.S., Lau, F.C.M.: Distributed $(\Delta + 1)$ Coloring in the Physical Model. Theoret. Comput. Sci. 553, 37–56 (2014)