# Oriented Edge Colorings and Link Scheduling in Sensor Networks*

Ted Herman Imran Pirwani Sriram Pemmaraju

## Abstract

*This paper shows that TDMA slot assignment for unicast communication in a wireless network can be distributively computed for an n-node acyclic network in $O(polylog(n))$ time, with high probability. The best previous distributed algorithm for this problem requires $O(n)$ time and obtains a TDMA schedule using $2\Delta$ time slots. The new algorithm uses $(1+\varepsilon)\cdot 2\Delta$ time slots where $\varepsilon$ is a tunable fraction.*

**Keywords:** edge-coloring, TDMA, wireless networks

## 1 Introduction

Power conservation is a priority in many wireless sensor networks. Basic activities of sensing and communication consume energy, and careful scheduling of operations can extend the useful lifetime of sensors and their batteries. Transmission is wasted in cases where receivers are not listening or where a receiver is the target of multiple transmissions (a collision). Time Division Multiple Access (TDMA) is a real-time scheduling technique to avoid such situations. Moreover, some popular sensor platforms consume as much power receiving as transmitting messages [14], so scheduling timeslots can also be used by receivers to further reduce energy needs by powering off the radio during idle periods.

Two communication modes of interest in sensor networks and ad hoc networks are local broadcast and unicast. A local broadcast transmission is the case where a node transmits a message to all its neighbors in the communication graph, and unicast is the case where a node transmits a message that is only intended for one neighbor. Local broadcast only succeeds if all the nodes neighbors receive the message without collision; unicast succeeds if (*i*) the targeted neighbor does not concurrently transmit, and (*ii*) the target receives without collision. Radio transmissions in sensor networks are not spatially directed. A unicast could be overheard by a number of the transmitter's neighbors; and at a *non-targeted* neighbor, there can be a collision or that neighbor could be transmitting concurrently. The success of the unicast depends only on reception at the target node. Thus the scheduling of unicast transmissions is less constrained than the scheduling of local broadcast transmissions.

If unicast is the communication mode of interest, then the problem of scheduling unicasts is equivalent to assigning to each edge in the communication network a color so that transmissions on edges of the same color are concurrent and there are no collisions at unicast destinations. This is the focus of our paper, TDMA for unicast. We refer to this as the *link scheduling* problem.

**Contributions of this paper.** The recent paper [6] proposes a distributed protocol to find a link schedule for acyclic networks. On an *n*-node network with maximum degree $\Delta$, the protocol runs in $\Theta(n)$ time and computes a link schedule using $2\Delta$ time slots. This paper is motivated by the fact that for large sensor networks, protocols with linear running time may be unacceptably slow. We present a simple, randomized, distributed protocol that runs in $O(\text{polylog}(n))$ time on acyclic networks and computes a link schedule using $2\Delta\cdot(1+\varepsilon)$ time slots, for any fraction $\varepsilon > 0$.

**Organization.** The remainder of the paper starts in Section 2 with a brief review of link scheduling on sensor networks. This is followed in Section 3 with a description of the connection between link scheduling and two graph coloring problems: distance-2 edge coloring and oriented edge coloring. Section 4 presents the main result of this paper: a simple, fast, randomized protocol for oriented edge coloring for acyclic networks, followed by its analysis. The oriented edge coloring constructed by our protocol yields a link schedule that uses close to optimal number of time slots. In Section 5 we report on experiments that provide additional insights into our results. The paper ends in Section 6 with a brief discussion of future work.

## 2 Transmission Scheduling in Sensor Networks

Protocols that schedule transmissions to avoid collision can be randomized, use static schedules, or be a hybrid of both [2]. All such protocols exploit, to varying degrees, access to real-time hardware in the participating nodes. Access to synchronized time in sensor networks can be an out-of-band signal, such as GPS or a standard time broadcast service [11], or generated by clock synchronization protocols now widely implemented for sensor networks [16]. Randomized protocols [17] reduce the probability of collisions, but do not eliminate collisions entirely; advantages of randomized protocols are simplicity and fault tolerance. Static schedules exclude all collisions, but require setup time and regeneration as nodes join and leave the network. Papers [10, 3, 8] are examples of proposed static scheduling in sensor networks. Hybrid protocols divide time into two phases, a contending phase (which can use randomized backoff), and a message phase. During the contending phase, participating nodes agree on a static schedule for the subsequent message phase.

Sensor networks differ from other wireless, ad hoc networks in that nodes have less powerful computing hardware, power conser-

*The authors are at the Department of Computer Science, The University of Iowa, Iowa City, IA 52240-1419. E-mail: [herman , pirwani , sriram ]@cs.uiowa.edu .

vation is crucial, and sensor network protocols cannot rely on powerful and complex base-station infrastructure (such as cellular networks do). In sensor networks, it is therefore a reasonable idea to use transmission scheduling layered on top of a randomized, lower-layer access protocol such as CSMA [17]: a TDMA schedule can have coarser granularity of time slots than the underlying CSMA protocol, and be used chiefly to conserve power, by turning off radios during periods when no communication is scheduled.

Sensor networks are an instance of ad hoc networking, where topology is dynamic. Moreover, sensor nodes may enter a sleeping state to harvest ambient energy, and later awake to rejoin the system. It follows that link scheduling is not a one-time problem. It is generally an open research problem how best to accommodate topology change in the context of TDMA. Depending on application needs and the frequency of join/leave events, periodic time slot reassignment can be adequate to deal with topology change. Periodically recomputing a link schedule is only viable if the overhead is low — this is an important motivator for our research, namely to find algorithms that quickly (in sublinear time) compute a link schedule, and do so distributively rather than having to rely on a centralized solution.

## 3 Coloring and Time Slots for Link Scheduling

The connection between transmission scheduling and graph coloring problems has been studied extensively [15], where scheduling parameters include frequency, time, and spatial domains. Here, we limit the investigation to scheduling in the time domain (TDMA) assuming reuse of a single radio frequency. Spatial reuse is a consequence of the fact that sensor networks have low power radios, enabling concurrent transmission by distant nodes without collision. As in several other previous works on TDMA in sensor networks, we make simplifying assumptions about the graph of communication: edges in the graph represent links with bidirectional communication. In practice, communication links may be unidirectional and even if two nodes $p$ and $q$ are unable to directly communicate, $p$ may be able to interfere with $q$'s reception of a transmission from another node. These situations, for purposes of TDMA slot assignment, can also be modeled by graph edges [10] which put constraints on allowable time slot assignments to avoid collision and interference. We consider these issues, as well as other abilities of sensor nodes to attenuate radio transmission power levels, to be outside the scope of our present investigation.

A *distance-k edge coloring* assigns colors to edges such that for any two edges $e$ and $e'$ with the same color, the minimum path length between $e$ and $e'$ is at least $k$. Two graph coloring problems can be exploited to find a link schedule, as follows. Suppose the edge colors are $1,2,\ldots,t$. Then assign discrete time slots $1,2,\ldots,2t$ so that color $i$ corresponds to time slots $2i-1$ and $2i$; for any edge $\{u,v\}$ of color $i$, time slot $2i-1$ is reserved for transmission from $u$ to $v$, and time slot $2i$ is reserved for transmission from $v$ to $u$. Distance-$j$ edge colorings for $j > k$ generally require the use of more colors than distance-$k$ edge colorings, and a requirement for a larger number time slots decreases the frequency at which a given vertex is allowed to transmit. As observed in [6], there are cases where distance-1 edge coloring (also called *proper edge coloring*) can be used to solve link scheduling, such as for acyclic graphs. Proper edge coloring can only be used if colors are "oriented", as defined below.

**Distance-2 Edge Coloring.** Let $G=(V,E)$ be a graph, with $\Delta(G)$ the maximum vertex degree. A *matching* $M$ of $G$ is a subset of edges such that no two edges in $M$ are incident on the same vertex. A *proper edge coloring* of $G$ is as assignment of colors to the edges of $G$ such that each color class is a matching. The *chromatic index* of $G$, denoted $\chi_e(G)$, is the fewest number of colors used in any proper edge coloring of $G$. Clearly, $\Delta(G) \leq \chi_e(G)$ and Vizing's theorem tells us that $\chi_e(G) \leq \Delta(G)+1$. Let EC denote the problem of computing an edge coloring of a given graph $G$ using the fewest number of colors. EC is NP-complete [9], despite the fact that $\chi_e(G) \in \{\Delta(G), \Delta(G)+1\}$, and despite the fact that the proof of Vizing's theorem yields a simple algorithm to color $G$ using $\Delta(G)+1$ colors [13].

A *distance-2 matching* $M$ of $G$ is a subset of edges such that no two edges in $M$ are incident on the same vertex and no two edges in $M$ are incident on neighbors. Such a matching is also called an *induced matching* [4, 5]. A *distance-2 edge coloring* of $G$ is a proper edge coloring of $G$ in which each color class is a distance-2 matching. Let D2EC denote the problem of computing a distance-2 coloring with the least number of colors possible. D2EC is in general harder than EC and is in fact NP-complete even for certain subclasses of bipartite graphs [12]. Let $\chi_e^2(G)$ denote the *distance-2 edge chromatic number* of $G$, the fewest colors needed by a distance-2 coloring of $G$. Part of the difficulty in solving D2EC comes from the fact that $\chi_e^2(G)$ can range from $\Delta(G)$ to $\Theta(\Delta(G)^2)$. Given the negative results regarding D2EC , it is reasonable to seek efficient approximation algorithms for this problem. [1] reports on constant-factor approximation algorithms for D2EC for planar graphs and unit disk graphs (UDGs).

**Oriented Edge Coloring.** An *orientation* of an undirected graph is an assignment of a direction to each edge. If $O$ is an orientation of an undirected graph $G$, then for each $e=\{u,v\} \in E(G)$, we have that $O(e) \in \{(u,v),(v,u)\}$. If $O(e)=(u,v)$ then we say that $u$ is the *tail* of $e$ and $v$ is the *head* of $e$. An *oriented edge coloring* of a graph $G$ is an ordered pair, $(C,O)$ consisting of a proper edge coloring $C$ of $G$ and an orientation $O$ of $G$ such that for any two edges $e_1=\{u_1,v_1\}$ and $e_2=\{u_2,v_2\}$ that are incident on neighbors and have the same color, the orientation $O$ satisfies the following condition:

> for any $x \in e_1$ and $y \in e_2$, if $x$ and $y$ are neighbors, then either both $x$ and $y$ are tails of $e_1$ and $e_2$ respectively or both $x$ and $y$ are heads of $e_1$ and $e_2$ respectively.

See Figure 1 for an illustration of this definition.

An oriented edge coloring $(C,O)$ of $G$ that uses $t$ colors leads to a link schedule that uses $2t$ time slots. For each $i$, $1 \leq i \leq t$, assign time slot $2i-1$ and $2i$ to all edges colored $i$ with the requirement that for each edge $e=\{u,v\}$ colored $i$, the tail of $O(e)$ transmits to the head of $O(e)$ in slot $2i-1$ and the head of $O(e)$ transmits to the tail of $O(e)$ in slot $2i$.

Let $\vec{\chi}_e(G)$ denote the *oriented edge chromatic number* of $G$, the fewest colors needed by an oriented coloring of $G$. It is easy to see that $\vec{\chi}_e(G) \leq \chi_e^2(G)$. start with any distance-2 edge coloring $C$

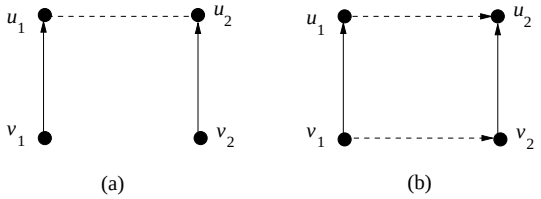


**Figure 1. Figures (a) shows an oriented edge coloring of a 3-path. No orientation is shown for edge $\{u_1,u_2\}$ because it does not matter. The edges $\{u_1,v_1\}$ and $\{u_2,v_2\}$ are assigned the same color and therefore they must be oriented as shown in (a) or both edges must be reversed. Figure (b) shows an oriented edge coloring of a 4-cycle using 2 colors. This implies a collision-free assignment of 4 time slots to the edges of the 4-cycle. Note that in any distance-2 edge coloring of a 4-cycle, all edges need to be assigned distinct colors, implying a collision-free assignment of 8 time slots to the edges.**

of $G$ and arbitrarily orient the edges of $G$ to get a valid oriented edge coloring of $G$. In fact, [6] show that every tree $T$ has an oriented edge coloring that uses $\Delta(T)$ colors. In contrast, it is easy to construct a tree $T$ that requires $2\Delta(T)-1$ colors in any distance-2 edge coloring. Link schedules constructed from oriented colorings typically use fewer colors than those constructed from distance-2 edge colorings - a fact that makes oriented colorings attractive.

**Distributed Edge Coloring.** The problem of devising distributed algorithms, for the problem of computing a proper edge coloring with a small number of colors, has received considerable attention. It is of course possible to construct a distributed version of Vizing's algorithm that produces a $\Delta+1$-coloring for any graph with maximum degree $\Delta$ [13]. This is essentially the technique used by [6] to obtain a $\Delta$-coloring of the edges of an acyclic graph. The problem with this approach is that it runs in linear time.

There are several randomized, distributed algorithms that run in $O(\text{polylog}(n))$ time on $n$-vertex graphs and produce an edge coloring close to an optimal edge coloring. Currently, the fastest distributed edge coloring algorithm is due to Grable and Panconesi [7] that computes a proper edge coloring of a given graph $n$-vertex graph $G$ in $O(\text{polylog}(n))$ rounds using $(1+\varepsilon)\Delta(G)$ colors for any $\varepsilon > 0$. This algorithm is also extremely simple and we reproduce it here. Each edge $e=\{u,v\}$ is initially given a palette of $(1+\varepsilon)\max\{deg(u),deg(v)\}$ colors. The computation takes place in rounds. In each round, each uncolored edge independently picks a tentative color uniformly at random from its current palette. If no neighboring edge picks the same color, it becomes final. Otherwise, the edge tries again in the next round. At the end of each round palettes are updated: colors successfully used by neighboring edges are deleted from the current palette. This algorithm is inherently distributed since each edge only needs to exchange information with its neighboring edges. We call this the Grable-Panconesi algorithm and use it later.

**Distributed Edge Orientation.** The *line graph* of a graph $G=(V,E)$, denoted $L(G)$ has vertex set $E$ and edges connecting $e_1,e_2 \in E$, whenever $e_1$ and $e_2$ share an endpoint. The problem of edge coloring $G$ is equivalent to the problem of vertex coloring $L(G)$. The *square* of a graph $G=(V,E)$, denoted $G^2$ has vertex set $V$ and edges connecting pairs of vertices that are at distance at most 2 from each other in $G$. The problem of distance-2 edge coloring $G$ is equivalent to the problem of vertex coloring $L^2(G)$.

Let $T$ be a tree and $C$ be a proper edge coloring of $T$. For any color $c \in C$, let $L^2(T,c)$ denote the subgraph of $L^2(T)$ induced by edges colored $c$. See Figure 2 for an illustration. The following

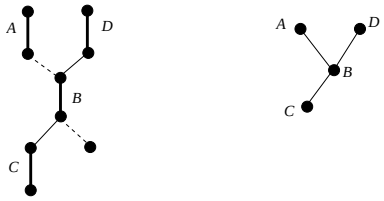


**Figure 2. A 3-edge coloring of a tree $T$ is shown on the left. On the right $L^2(T,c)$ is shown, where $c$ is the color assigned to the thick edges.**

proposition is easy to verify.

**Proposition 1** *For any tree $T$, any proper edge coloring $C$ of $T$, and any color $c$ used by $C$, $L^2(T,c)$ is a forest.*

[6] presents an algorithm which starts with a proper edge coloring $C$ of $T$ and then constructs an orientation $O$ of $T$ such that $(C,O)$ is a valid oriented edge coloring of $T$. This is done by doing a traversal (depth-first) of $L^2(T,c)$ for each color $c \in C$. Given that $L^2(T,c)$ is a forest, the algorithm picks a component of $L^2(T,c)$, makes an arbitrary node the root, assigns the root an arbitrary orientation and then repeatedly assigns an orientation to children that is consistent with their parent's orientation (technically, all vertices begin as candidates for the root, initiating multiple traversals in parallel; the traversal initiated at the vertex with maximum identifier overtakes all other traversals to get the final orientation). The worst case running time of this algorithm is $\Theta(m)$, where $m$ is the number of vertices in $L^2(T,c)$. In the worst case, this can be linear in number of vertices of $T$. Since the line graphs for each color can be oriented independently, $|C|$ concurrent incarnations of the traversal can orient all the edges in $O(n)$ time.

## 4 A distributed algorithm for oriented coloring

We propose a randomized algorithm that takes an $n$-vertex tree $T$ with maximum degree $\Delta$, and for any fraction $\varepsilon > 0$, returns an oriented edge coloring $(C',O)$ of $T$, where $C'$ uses at most $(1+\varepsilon)\cdot\Delta$ colors. The algorithm runs in $O(\text{polylog}(n))$ time, with the asymptotic notation hiding a constant that depends on $\varepsilon$.

Execution of the algorithm proceeds in three phases. The first phase uses the randomized Grable-Panconesi algorithm to obtain a proper edge coloring. The second phase consists of a randomized recoloring of the edges using an expanded palette. The edge coloring remains proper after this phase. The third phase orients edges of each color.

Algorithm `OrientedEdgeColoring` $(T, \varepsilon)$

1. Use the Grable-Panconesi algorithm to compute a proper edge coloring $C$ of $T$ with $\Delta(1+\varepsilon/3)$ colors.
2. `ExpandPalette` $(T, C, \varepsilon)$.
3. Let $C'$ be the expanded set of colors used for the edges of $T$. For each color $c \in C'$, orient the edges colored $c$ by constraint propagation in $L^2(T,c)$.

The first phase completes in $O(\text{polylog}(n))$ time and so if we can complete the remaining two phases also in $O(\text{polylog}(n))$ time, then the entire algorithm would run in $O(\text{polylog}(n))$ time. But, how can the orientation phase be run in $O(\text{polylog}(n))$ time? The depth-first traversal technique used for orientation in [6] takes time proportional to the number of vertices in $L^2(T,c)$, and this can, in the worst case be linear in the number of vertices in $T$. We make our first improvement by replacing depth-first traversal with a constraint propagation technique, whose time complexity is proportional to the diameter of $L^2(T,c)$ rather than the number of vertices. We sketch the details now. An immediate consequence of the definition of an edge orientation is: if $OR$ is an orientation, then $\overline{OR}$ is also an orientation, where $\overline{OR}$ is obtained by reversing the orientation of $OR(e)$ for every $e \in E$. Therefore, any chosen "root" vertex $r$ of a component of $L^2(T,c)$ can arbitrarily select some orientation. The orientation selected by $r$ is a sufficient constraint to fix the orientations of all neighbors of $r$ in $L^2(T,c)$; these neighbor orientations can be assigned concurrently in $O(1)$ time. It follows by induction that all edges of each component of $L^2(T,c)$ will be oriented within time proportional to the maximum diameter of any component of $L^2(T,c)$. We call this procedure the *constraint propagation* algorithm for orientation. As with the traversal algorithm used in [6], there is no need to explicitly establish a root vertices in each component of $L^2(T,c)$. All vertices start, in parallel, to initiate constraint propagation. Orientations are tagged by the identity of the root constraint; constraint propagation initiated by a vertex with a larger identity overtakes those with smaller identities.

Thus the running time of the third phase is proportional to the maximum diameter of any component of $L^2(T,c)$, for any color $c$. In the rest of this section we present the second phase, which is a randomized recoloring of the edges of $T$ with an expanded palette, and show that after this phase, the maximum diameter of any monochromatic component of $L^2(T)$ is $O(\log n)$, with high probability.

We use the following notation in the algorithm. Let $k$ be an integer exceeding $3/\varepsilon$ and let $p = 1/2$. Partition the set $\{1,2,\ldots,t\}$ of colors used by $C$ into subsets $C_1, C_2, \ldots, C_\ell$, where $\ell = \lceil t/k \rceil$, and

$$\begin{aligned} C_i &= \{k(i-1)+1, k(i-1)+2, \ldots, k\cdot i\}, \text{ for } 1 \le i \le \ell-1, \\ C_\ell &= \{k(\ell-1)+1, k(\ell-1)+2, \ldots, t\} \end{aligned}$$

For each $i$, $1 \le i \le \ell$, let $E_i$ denote the edges of $T$ colored using some color in $C_i$.

Algorithm `ExpandPalette` $(T, C, \varepsilon)$

1. For each $i$, $1 \le i \le \ell$, for each edge $e \in E_i$, recolor $e$ with a new color, $(t+i)$, with probability $p$.
2. For each vertex $v$ in $T$ and for each $i$, $1 \le i \le \ell$, there may be up to $k$ edges incident on $v$ that are colored using $t+i$. If there are two or more edges incident on $v$ that are both colored $t+i$, then pick one of these edges uniformly at random and retain its new color. For the rest of the edges incident on $v$ and colored $t+i$, restore the color of each of these edges to their original color.

Let $C'$ be the final edge coloring of $T$ and let $O$ be the orientation of $T$ computed by the algorithm. It is easy to see that the algorithm guarantees that $(C', O)$ is an oriented edge coloring of $T$. Let $t$ be the number of colors used by the Grable-Panconesi algorithm. Note that $C'$ uses $t+\ell$ colors. Given that $k > 3/\varepsilon$, it follows that $\ell = \lceil t/k \rceil < \lceil t\varepsilon/3 \rceil$, implying that $\ell \le t\varepsilon/3$. Therefore, $C'$ uses at most $t\cdot(1+\varepsilon/3)$ colors. Since $t \le \Delta\cdot(1+\varepsilon/3)$, the total number of colors used by $C'$ is at most $\Delta\cdot(1+\varepsilon/3)^2$. If $0 < \varepsilon < 1$, then $(1+\varepsilon/3)^2 \le (1+\varepsilon)$ and we get the following result.

**Proposition 2** *$(C', O)$ is an oriented edge coloring of $T$ using at most $\Delta\cdot(1+\varepsilon)$ colors.*

We now show that with high probability, there is no connected component of $L^2(T,c)$ for any $c$, whose diameter exceeds $a\log_2 n$ for some constant $a$ (which depends on $\varepsilon$). This shows that with high probability, phase 3 of `OrientedEdgeColoring` runs in $O(\log n)$ communication rounds. Therefore the entire algorithm runs in $O(\text{polylog}(n))$ time with high probability.

**Theorem 3** *Let $c$ be a color used by $C'$. There is a constant $a = a(\varepsilon)$ such that the probability that the diameter of a connected component of $L^2(T,c)$ exceeds $a\log_2 n$, is at most $1/n$.*

**Proof:** Fix an edge $e$ of $T$ and let $c$ be a color such that $1 \le c \le t$. Thus $c$ is one of the original colors used for the edges of $T$. We first compute the probability that $e$ is colored $c$ at the end of `ExpandPalette`. If $e$ is not colored $c$ originally, that is, before the start of the algorithm, then the probability that $e$ is colored $c$ by the algorithm is 0. So we assume that $e$ is originally colored $c$. In this case $e$ continues to be colored $c$ either because $e$ was never recolored or because $e$ was recolored and then restored to its original color in Step 2. Therefore,

$$\text{Prob}[e \text{ is colored } c] \le (1-p) + p\cdot\frac{k-1}{k} = 1 - \frac{p}{k}.$$

Now we compute the probability that an edge $e$ is colored $c$, where $c$ is a new color, that is, $t < c \le t+\ell$. For $e$ to have a new color when `ExpandPalette` ends, it must be the case that it was recolored in Step 1 and retained its new color in Step 2. The upper bound on this probability is $p$. Since $p = 1/2$, the quantity $1 - p/k \ge p$ for all $k \ge 1$. Thus we conclude that for any edge $e$ and any color $c$, $1 \le c \le t+\ell$,

$$\text{Prob}[e \text{ is colored } c] \le 1 - \frac{1}{2k}.$$

For notational convenience we denote the quantity $1 - 1/2k$ by $\beta$.

Now consider a path $P$ in $L^2(T)$ of length $L$. Fix a color $c$, $1 \leq c \leq t + \ell$. We calculate the probability that $P$ appears in $L^2(T,c)$. For this to happen every vertex in $P$ needs to be colored $c$. Noting that vertices in $P$ are colored $c$ independently, we obtain

$$\text{Prob}[P \text{ is in } L^2(T,c)] \leq \beta^L.$$

We now compute the probability that for some $c$, there is a path in $L^2(T,c)$ of length exceeding $L$. Let $\mathcal{P}$ be the collection of all paths in $L^2(T)$ of length greater than $L$. The probability we are interested in computing is the probability that there is a path $P \in \mathcal{P}$ that appears in $L^2(T,c)$ for some $c$. Using the union bound, we see that this quantity is bounded above by

$$\begin{aligned} \sum_c \sum_{P \in \mathcal{P}} \text{Prob}[P \text{ appears in } L^2(T,c)] &\leq \sum_c \sum_{P \in \mathcal{P}} \beta^L \\ &\leq n^3 \cdot \beta^L \end{aligned}$$

The last inequality follows from the fact that (i) $L(T)$ is a forest with fewer than $n$ vertices and there are fewer than $n^2$ distinct paths in such a graph and (ii) the algorithm uses at most $n$ colors.

Now substituting $L = a \log_2 n$ in the above upper bound, we get that the probability that there is path in $L^2(T,c)$ for some $c$, of length greater than $L$ is bounded above by

$$n^3 \cdot \beta^L = n^3 \cdot \beta^{a \log_2 n} = n^3 \cdot n^{a \log_2 \beta} = \frac{n^3}{n^{a \log_2(1/\beta)}}.$$

If we let $a = 4/log_2(1/\beta)$, then the above upper bound simplifies to $1/n$. Thus we have that the probability that there is a connected component of $L^2(T,c)$ for some $c$ whose diameter exceeds $a \log_2 n$ is bounded above by $1/n$. □

## 5 Experimental Results

We performed experiments to measure the maximum diameter of monochromatic components of $L^2(T)$ before and after recoloring the tree $T$ with extra colors to get a sense of how much reduction in the maximum diameter is achieved by expanding the palette. We ran our simulations on three different types of trees. These experiments were done using the *Combinatorica* package that comes with *Mathematica 5.2*. In the following, we summarize our experiments and resulting observations. For more details, including raw data, see `www.cs.uiowa.edu/~pirwani/TDMA/sensorware2006`.

We generated three types of trees.

(i) **Random trees**. Given a positive integer $n$, we construct an $n$-vertex labeled tree picked uniformly at random from the collection of all $n$-vertex labeled trees. Note that the resulting probability distribution is not uniform on unlabeled trees.

(ii) **Caterpillar trees**. Given positive integers $n$ and $\Delta$, we construct an $n$-vertex tree with maximum vertex degree $\Delta$ as follows. Start with a path of length $k$. To each of the $k-2$ internal vertices of the path we connect $\Delta - 2$ new vertices and to the two endpoints of the path we connect $\Delta - 1$ new vertices. The value $k$ is chosen so that the total number of vertices of the tree equal $n$. Note that $k$ is roughly equal to the diameter of the tree. Caterpillar trees are particularly bad for proper edge colorings that use $\Delta$ colors in the sense that every maximal monochromatic subgraph of $L^2(T)$ is a long chain of length approximately $k$. Our expectation is that for caterpillar trees, the benefits of expanding the palette would be significant.

(iii) **Bush trees**. A *bush tree* is constructed by starting with a random tree $T$ and connecting each internal vertex $u$ to $\Delta(T) - degree(u)$ new vertices.

A typical experiment consisted of generating an $n$-vertex tree $T$ with maximum degree $\Delta$, constructing a $\Delta$-edge coloring of $T$, and then running `ExpandPalette`. 20 iterations of this experiment were performed for each tree type and averages taken over these 20 runs. We reported the maximum diameter of a monochromatic component of $L^2(T)$ before and after running `ExpandPalette`. These are shown in the last two rows of the table below. The reduction in the diameter is significant in all cases and as expected, quite dramatic in the case of the caterpillar tree.

| | Random | Caterpillar | Bush |
|---|---|---|---|
| $n$ | 1000 | 1000 | 1237.75 |
| $\Delta$ | 6.4 | 9 | 5.9 |
| *diam* | 97.45 | 126 | 62.35 |
| Old color usage | 62.80% | 62.05% | 63.41% |
| Initial max. diameter | 58.05 | 124 | 60 |
| Final max. diameter | 13 | 16 | 16 |

The row labeled "Old color usage" reports the fraction of edges that are colored using an old color (that is, a color in the range 1 through $\Delta$). These fractions are remarkably similar for all three tree types and furthermore they are roughly equal to the fraction of colors in the final coloring that are old (66%). Trying to keep the usage of old colors equal to the fraction of old colors seems to be a reasonable heuristic for achieving substantial reduction in the diameter of monochromatic components. For example, when we used $p = 2/3$, the average number of edges colored using old colors is about 55%. This decrease in the usage of the old colors implies that the new colors are being overused. This imbalance corresponds to a smaller reduction in the diameter of monochromatic components at least in the case of the Caterpillar and the Bush trees, as shown below.

| | Random | Caterpillar | Bush |
|---|---|---|---|
| Initial max. diameter | 58.02 | 124 | 58.35 |
| Final max. diameter | 11.7 | 32.25 | 22 |

We also implemented the Grable-Panconesi algorithm for proper edge coloring. Our main aim in implementing this algorithm was to see if the Grable-Panconesi algorithm, by itself leads to small diameter monochromatic components. The table below (last two rows) shows that for all three tree types, just running the Grable-Panconesi algorithm leads to smaller monochromatic components than starting with a $\Delta$-edge coloring and applying `ExpandPalette`. These experimental results strongly motivate the following question. Let $C$ be a Grable-Panconesi edge coloring of a tree $T$ using $\Delta(1+\varepsilon)$ colors, for some $\varepsilon > 0$. Is the maximum diameter of a component in $L^2(T,c)$, for all colors $c$, bounded above by $O(\log n)$ with high probability? Analysis of the

Grable-Panconesi algorithm for obtaining bounds on the diameters of monochromatic components, seems more difficult than the analysis in this paper because of dependencies between the coloring events being analyzed. However, this is an important question for the future.

| | Random | Caterpillar | Bush |
|---|---|---|---|
| $n$ | 1000 | 994 | 1285 |
| $\Delta$ | 6.65 | 8 | 6.15 |
| *diam* | 96.25 | 143 | 58.65 |
| Grable-Panconesi | 8.3 | 13 | 15.1 |
| `ExpandPalette` | 12.8 | 16.25 | 15.7 |

## 6 Discussion

The analysis of the `ExpandPalette` algorithm crucially used the fact that the number of distinct paths in an $n$-vertex tree is polynomial in $n$. This is false for most other classes of graphs. As a consequence the analysis does not directly apply to other classes of graphs. However the overall three phase scheme of computing a proper edge coloring, expanding the palette, and then orienting edges in each monochromatic component, may be useful for other classes of graphs. ([6] observes that line graphs for which all cycles are even in length have proper edge orientations, and expanding the palette could eliminate line graphs with odd-length cycles.) Motivated by this overall goal, we are interested in obtaining a more sophisticated probabilistic analysis of `ExpandPalette` and a possibly more sophisticated version of the edge orientation phase.

## References


[1] C L Barrett, G Istrate, V S Anil Kumar, M V Marathe, S Thite, and S Thulasidasan. Efficient algorithms for channel assignment in wireless radio networks. unpublished manuscript, 2005.

[2] D Bertsekas and R Gallager. *Data Networks*. Prentice-Hall, 1987.

[3] C Busch, M Magdon-Ismail, F Sivrikaya, and B Yener. Contention-free MAC protocols for wireless sensor networks. In *Proceedings of the 18th Annual Conference on Distributed Computing (DISC)*, pages 245–259, 2004.

[4] K Cameron. Induced matchings. *Discrete Applied Mathematics*, 24:97–102, 1989.

[5] R J Faudree, A Gýarfas, R H Schelp, and Zs Tuza. Induced matchings in bipartite graphs. *Discrete Math.*, 78:83–87, 1989.

[6] S Gandham, M Dawande, and R Prakash. Link scheduling in sensor networks: Distributed edge coloring revisited. In *INFOCOM*, 2005.

[7] D A Grable and A Panconesi. Nearly optimal distributed edge coloring in $O(\log\log n)$ rounds. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, 1997.

[8] T Herman and S Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *Proceedings of the First Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors'2004)*, pages 45–58. Springer LNCS 3121, 2004.

[9] I Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981.

[10] SS Kulkarni and U Arumugam. Collision-free communication in sensor networks. In *Proceedings of Self-Stabilizing Systems, 6th International Symposium*, pages 17–31. Springer LNCS 2704, 2003.

[11] M Lombardi, A Novick, J Lowe, M Deutch, G Nelson, D Sutton, W Yates, and D Hanson. WWVB radio controlled clocks: Recommended practices for manufacturers and consumers. Technical Report 960-14, United States National Institute of Standards, January 2005.

[12] M Mahdian. On the computational complexity of strong edge coloring. *Discrete Applied Mathematics*, 118:239–248, 2002.

[13] J Misra and D Gries. A constructive proof of vizing's theorem. *Information Processing Letters*, 41:131–133, 1992.

[14] J Polastre, R Szewczyk, and D Culler. Telos: enabling ultra-low power wireless research. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks; Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, 2005.

[15] S Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5(2):81–94, 1999.

[16] B Sundararaman, U Buy, and AD Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3:281–323, 2005.

[17] A Woo and D Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 221–235, 2001.