University of Rhode Island
DigitalCommons@URI

Open Access Dissertations

2011

RELAXATIONS OF L(1, 1)-LABELING FOR THE BROADCAST SCHEDULING PROBLEM

Shaun N. Joseph
University of Rhode Island, snjoseph@gmail.com

Follow this and additional works at: https://digitalcommons.uri.edu/oa_diss

Recommended Citation

Joseph, Shaun N., "RELAXATIONS OF L(1, 1)-LABELING FOR THE BROADCAST SCHEDULING PROBLEM" (2011). *Open Access Dissertations.* Paper 72.
https://digitalcommons.uri.edu/oa_diss/72

This Dissertation is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Dissertations by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons@etal.uri.edu.

RELAXATIONS OF $L(1,1)$-LABELING FOR THE BROADCAST SCHEDULING PROBLEM

BY

SHAUN N. JOSEPH

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

APPLIED MATHEMATICAL SCIENCES

UNIVERSITY OF RHODE ISLAND

2011

DOCTOR OF PHILOSOPHY DISSERTATION

OF

SHAUN N. JOSEPH

APPROVED:

Dissertation Committee:

Major Professor Lisa DiPippo

Edmund Lamagna

Nancy Eaton

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2011

ABSTRACT

The broadcast scheduling problem asks how an arbitrary network of broadcast transceivers operating on a shared medium may share the medium in such a way that communication over the entire network is possible. In the case where transmissions are explicitly scheduled, as opposed to be determined by contention, the problem is naturally modeled as a graph coloring problem. The canonical model is the $L(1,1)$-labeling, also known as the distance-2 coloring, coloring of the graph square, or strict schedule. This coloring is, however, difficult to obtain even sub-optimally and typically uses many colors, which corresponds to an undesirable over-division of the medium.

This work introduces a relaxation of $L(1,1)$-labeling called $\widetilde{L}(1,1)$-labeling or the pseudo-schedule. Whereas strict schedules guarantee that every path in the graph is a communication path, pseudo-schedules only require the existence of a communication path between any two vertices. The study shows that pseudo-schedules have many superior characteristics to the canonical model, provided the relaxation is acceptable. In particular, the worst case number of colors used is linear in the degree of the graph, as opposed to quadratic for strict schedules.

The formal properties of the $\widetilde{L}(1,1)$-labeling are comprehensively treated, including investigations of its "chromatic number," rigorous analysis of several algorithms, and proofs of hardness of optimization and approximation. Basic results on a generalization of the coloring are obtained, and nine open problems are posed for future research.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Lisa DiPippo, as well as the faculty, staff, and students of the Department of Computer Science and Statistics.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGMENTS iii

TABLE OF CONTENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1 The Broadcast Scheduling Problem 1

1.1 Explicit scheduling and graph coloring 3

1.2 Prerequisites 5

List of References 5

2 Pseudo-Schedules 7

2.1 Definition 7

2.1.1 Comparison to other labelings 9

2.2 Relations between the chromatic numbers 11

2.3 Bounds on $\widetilde{\chi}_{1,1}$ 12

2.3.1 Graphs for which $\widetilde{\chi}_{1,1}$ is known 13

2.4 Bounds on $\widetilde{\chi}_{1,1}$ for almost every graph 16

List of References 21

3 Algorithms 23

3.1 Spanning tree pseudo-schedules 23

Page

3.2 The greedy algorithm 25

3.2.1 Proof of correctness 25

3.2.2 Analysis 26

3.2.3 Implementation issues 28

3.3 The 2Δ algorithm 29

3.3.1 Proof of correctness 30

3.3.2 Analysis 31

3.3.3 Implementation issues 33

3.4 The d-band algorithm 33

3.4.1 Proof of correctness 36

3.4.2 Analysis 42

3.4.3 Implementation issues 43

3.5 Dismantleable (cop-win) graphs 44

3.5.1 Proof of correctness 45

3.5.2 Analysis 46

List of References 47

4 Hardness 52

4.1 Hardness of optimization 53

4.2 Hardness of approximation 60

List of References 61

5 Pseudo-(h, k)-Labelings 62

5.1 Bounds on $\widetilde{\chi}_{h,k}$ 63

5.2 Hardness 64

Page

5.2.1 Hardness of $\text{PLABEL}_{h,1}$ 65

List of References 73

6 Conclusion 74

6.1 Open problems 74

List of References 78

BIBLIOGRAPHY 79

LIST OF TABLES

Table		Page
1	Summary comparison of the main algorithms	23
2	Empirical performance of the greedy algorithm on random graphs	29

LIST OF FIGURES

Figure		Page
1	The hidden and exposed terminal problems	1
2	The RTS/CTS scheme	2
3	A pseudo-schedule on C_4	9
4	A graph with $\widetilde{\chi}_{1,1} = 3 < \chi = 4$	12
5	A pseudo-schedule that is not spanning tree	24
6	Degree-2 degenerate case for the greedy algorithm	27
7	A graph for which the 2Δ algorithm hits the upper bound	32
8	The 3-band algorithm on the 3-cube Q_3	36
9	Resolving of a dependency cycle of mixed type	39
10	Structure of $G(\mathcal{F})$	56

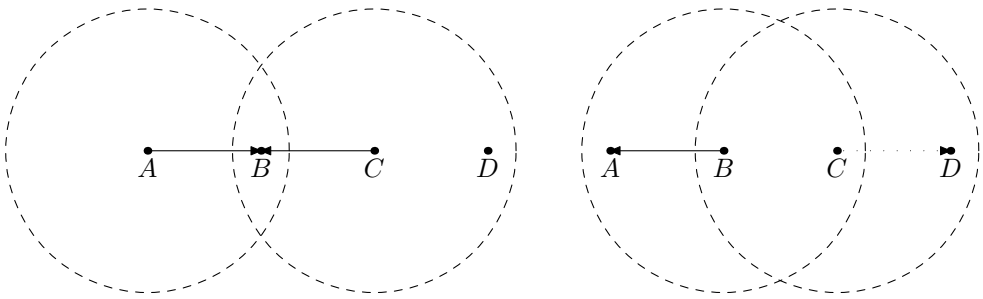
CHAPTER 1

The Broadcast Scheduling Problem

The *broadcast scheduling problem* asks how an arbitrary network of broadcast transceivers operating on a shared medium may share the medium in such a way that communication over the entire network is possible. In particular, two or more transmissions made simultaneously on the same medium should be expected to fail; ie, the transmissions conflict. The classic obstacles raised by the broadcast scheduling problem are the *hidden terminal problem* and the *exposed terminal problem*.

The hidden terminal problem posits that a node may seize the medium too aggressively, as it is generally not able to sense every other node with which it may conflict. Contrarily, the exposed terminal problem shows that a node may be too conservative in taking the medium, since not every node it senses necessarily engenders a conflict. Figure 1 illustrates the problems.

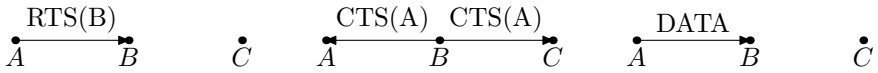
A *medium access control (MAC) protocol* is a practical solution to the broadcast scheduling problem. The predominant approach to MAC protocol design is *contention*, the outstanding example of which is carrier sense multiple access



(a) A and C sense transmissions from B, but not each other, so A and C may transmit simultaneously to B without effect

(b) C senses transmissions from B but only wants to transmit to D, so simultaneous transmission should be permitted

Figure 1: The hidden and exposed terminal problems, respectively.



(a) A requests permission to send to B

(b) B broadcasts clearance for A to send; C defers transmission

(c) A transmits data to B

Figure 2: The RTS/CTS scheme; in the event of simultaneous RTSs to the same node, no CTS will be issued, and the requesters will enter a probabilistic "back-off" routine.

(CSMA); examples include the wireless Ethernet standard 802.11 [1] and the protocol B-MAC [2] for wireless sensor networks. The hidden and exposed terminal problems are avoided through the use of a "request to send/clear to send" (RTS/CTS) scheme, as illustrated in Figure 2.

The performance of contention-based protocols is known to degrade seriously under high-traffic conditions, especially in the common case of sink-oriented networks, as demonstrated by Ahn et al. [3]. Furthermore, real-time applications may require predictability that the contention paradigm is unable to provide, since contention is inherently non-deterministic. Finally, contention imposes a high duty-cycle on the transceiver, since it must be ready to hear and respond to messages more or less uninterruptedly. This is a particularly undesirable property in wireless sensor networks, where transceivers are typically powered by battery.

The alternative to contention is *explicit scheduling*, the outstanding example of which is time division multiple access (TDMA). MAC protocols that (at least partially) integrate TDMA include (D)RAND [4], Z-MAC [5], Funneling-MAC [3], and DCQS [6]. In explicit scheduling, the hidden and exposed terminal problems can be avoided by construction of the schedule.

Explicit scheduling overcomes the aforementioned disadvantages of contention but introduces its own. First, determining a nontrivial explicit schedule is a difficult problem, especially in ad-hoc networks. Second, and relatedly, a schedule may

be too "large" in the sense that it demands too much division of the medium; for example, a TDMA schedule may have an excessive number of timeslots per frame, imposing a long pause between subsequent transmissions from the same node. Finally, the transceivers must be synchronized in the sense of adhering to the schedule; thus a node under TDMA must only transmit during its assigned timeslot, and nodes must have synchronized clocks.

This study will address the first two issues by offering a new formal framework, the $\widetilde{L}(1,1)$-labeling or pseudo-schedule, in which to consider the construction of explicit schedules for broadcast networks. (The synchronization problem is extremely interesting, but several solutions exist; see the survey [7].) Our treatment will be almost entirely mathematical, albeit motivated by the practical concerns just discussed, and with practical applications that are easy to discern.

The remainder of this chapter briefly introduces the concept of modeling explicit scheduling as a graph coloring problem, the canonical $L(1,1)$-labeling model, and why one would want to relax this model. These points are pursued rigorously in §2, wherein is defined the pseudo-schedule, and its fundamental properties analyzed. In §3 we encounter and analyze several pseudo-scheduling algorithms, and in §4 we prove that no efficient optimal algorithms, nor in fact a certain class of approximation algorithms, exist, unless P = NP. A natural generalization of pseudo-schedules, primarily of theoretical interest, is treated in §5, and we conclude the study in §6.

1.1 Explicit scheduling and graph coloring

Any network is easily modeled as a mathematical graph, with the network nodes as vertices and communication links as edges. The explicit scheduling of a network can then be seen as a kind of graph coloring, where each vertex is "colored" with the segment of the medium assigned to it by the schedule.

Assuming communication links are symmetric—as we will throughout the study—the broadcast scheduling problem is solved via an explicit schedule arising from a coloring of the corresponding graph such that any vertex is colored differently than any other vertex at distance one or two. Such a coloring is called an $L(1,1)$-labeling, or alternately a distance-2 coloring, coloring of the graph square, or strict schedule. This is, in fact, the canonical model for explicit scheduling, and is implemented as a MAC protocol by (D)RAND.

Unfortunately $L(1,1)$-labeling leads to "large" schedules: the division of the medium grows as the square of the maximum number of communication links at any node in the network. This is essentially because $L(1,1)$-labeling completely bars transmission conflicts, even if such conflicts would not disconnect the network. A network with a routing tree, for example, only needs to guarantee lack of transmission conflicts on the tree—it is a matter of indifference whether off-tree links are valid transmission vectors, since they are never used. In this sense, $L(1,1)$-labeling "over-solves" the broadcast scheduling problem.

It is natural, then, to ask how one might solve the broadcast scheduling problem while permitting certain "harmless" conflicts. This is not a completely novel idea. Z-MAC and Funneling-MAC mix TDMA and CSMA with the notion that a network may benefit from opportunistically choosing between them. In a related but distinct vein, TSMA [8] and RIMAC [9] allow conflicts in explicit schedules while attempting to make some probabilistic guarantees about data delivery.

All the aforementioned protocols, however, abandon the determinism that constitutes one of the benefits of explicit scheduling. The pseudo-schedule, by contrast, is defined as a graph coloring, with deterministic and provable characteristics. For instance, it can be proved that no conflicts present in a pseudo-schedule actually disconnect the network.

Ren's RAC-CT [10] implements what we recognize as the greedy algorithm for pseudo-scheduling, although the analysis is only completed here (in §3.2). With a firmer formal foundation, we are also able to present and analyze two alternative algorithms with some superior characteristics.

1.2 Prerequisites

The thesis assumes familiarity with the conventional terms and notation of graph theory. Except where otherwise stated, we implicitly assume that graphs are *simple*, that is: unweighted, undirected, and without loops or multiple edges. Recall that for any graph G, we typically let $V(G)$ denote the set of vertices of G, and $E(G)$ the set of edges. There appears to be no universally-accepted notation for the neighborhood, so let the *(closed) neighborhood* of a vertex v in graph G, defined as the set comprised of v and all vertices adjacent to v in G, be denoted $N_G[v]$. Recall that $\deg_G(v) = |N_G[v]| - 1$ denotes the *degree* of v in G and that $\Delta_G = \max_{v \in V(G)} \deg_G(v)$ denotes the *degree* of G itself. Finally, recall that the *distance* between vertices u, v in G, denoted $\mathrm{dist}_G(u, v)$, is the number of edges in the shortest path between u and v (with $\mathrm{dist}_G(u, v) = \infty$ for u, v disconnected).

For §2.4 the reader ought to have a basic grasp of the so-called "probabilistic method," particularly as it relates to random graphs. The textbooks of Alon and Spencer [11] and Bollobás [12] are the definitive references, respectively.

Finally, §4–5 assume that the reader is fully conversant in the theory of NP-Completeness. Any modern text in the theory of computation will suffice as a reference.

All other terms and notation are introduced and defined as needed.

List of References

[1] IEEE 802.11 Working Group, *ANSI/IEEE Std 802.11*, 1999th ed., IEEE.

[2] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Second ACM Conference on Embedded Network Sensor Systems (SenSys 2004)*, November 2004, pp. 95–107.

[3] G. Ahn, E. Miluzzo, A. T. Campbell, S. Hong, and F. Curomo, "Funneling-MAC: A localized, sink-oriented MAC for boosting fidelity in sensor networks," in *Fourth ACM Conference on Embedded Network Sensor Systems (SenSys 2006)*, November 2006, pp. 293–306.

[4] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks," in *Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)*, May 2006, pp. 190–201.

[5] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a hybrid MAC for wireless sensor networks," in *Third ACM Conference on Embedded Network Sensor Systems (SenSys 2005)*, November 2005, pp. 90–101.

[6] O. Chipara, C. Lu, and J. Stankovic, "Dynamic conflict-free query scheduling for wireless sensor networks," in *Fourteenth IEEE International Conference on Network Protocols (ICNP '06)*, November 2006, pp. 321–331.

[7] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.

[8] I. Chlamtac, A. Faragó, and H. Zhang, "Time-spread multiple access protocols for multihop mobile radio networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 804–812, 1997.

[9] L. DiPippo, D. Tucker, V. Fay-Wolfe, K. L. Bryan, T. Ren, W. Day, M. Murphy, T. Henry, and S. Joseph, "Energy-efficient MAC for broadcast problems in wireless sensor networks," in *Third International Conference on Networked Sensing Systems*, June 2006.

[10] T. Ren, "Graph coloring algorithms for TDMA scheduling in wireless sensor networks," Ph.D. dissertation, University of Rhode Island, 2007.

[11] N. Alon and J. H. Spencer, *The Probabilistic Method*, 3rd ed., ser. Wiley-Interscience Series in Discrete Mathematics and Optimization. John H. Wiley & Sons, 2008.

[12] B. Bollobás, *Random Graphs*, 2nd ed., ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

CHAPTER 2

Pseudo-Schedules

Having motivated the study from practical considerations in the previous chapter, we are now ready to engage in formal work. In this fundamental chapter, we define the pseudo-schedule and examine its basic properties, in particular the bounds on its "chromatic" number $\widetilde{\chi}_{1,1}$. We obtain many minor or specialized results, but the main one is given below.

Theorem 2.1. *Given a graph G with degree $\Delta \geq 1$,*

$$\Delta^* + 1 \leq \widetilde{\chi}_{1,1}(G) \leq 2\Delta$$

where Δ^* is the smallest possible degree of a spanning tree of G.

Before continuing, let us note that we will often implicitly assume in our proofs that we work with connected graphs. There is no danger in this, since a disconnected graph may be taken component-by-component without loss of generality; the basic definitions have been written carefully to ensure this is so.

2.1 Definition

Let us first introduce in a very general way the "strict" structure that we wish to relax, with notation adapted from Calamoneri [1]. An $L(h_1, \ldots, h_k)$-*labeling* of graph G, where $h_1 \geq h_2 \geq \cdots \geq h_k \geq 1$, is a function $l : V(G) \to \mathbb{Z}$ such that $|l(u) - l(v)| \geq h_i$ if $\mathrm{dist}_G(u, v) = i$. If l has image $K \subset \mathbb{Z}$, the $(h_1, h_2, \ldots, h_k)$-*number* of G, denoted $\chi_{h_1,h_2,\ldots,h_k}(G)$, is the smallest possible value of $(\max K - \min K + 1)$. (Observe that when $h_1 = h_2 = \cdots = h_k = 1$, we can simply minimize $|K|$.) Clearly an $L(1)$-labeling of G is identical to a *proper coloring* of G, and the (1)-number $\chi_1(G)$ (or just $\chi(G)$) is the well-known *chromatic number*. For this

reason the range set of the labeling function is also known as the set of "colors," and we indifferently use the name "coloring" for labeling, even outside the case of $L(1)$. We also say that the $(h_1, h_2, \ldots, h_k)$-number is the chromatic number of the $L(h_1, \ldots, h_k)$-labeling.

An $L(1,1)$-labeling assigns colors to vertices such that no vertex has the same color as any other vertex within a radius of distance two. In §1 we discussed how this labeling, which is also called *distance-2 coloring*, *coloring the graph square*, or *strict scheduling*, is related to the broadcast scheduling problem.

We now shift perspective. Relative to some labeling function l as above, we say that the ordered pair $(u, v) \in V(G)^2$ is *nonconflicting* iff uv is an edge in G and for every $x \in N_G[v]$ distinct from u, $l(x) \neq l(u)$. (In a broadcast network, this is analogous to a node u transmitting to v without interference.) A directed path from u to v is likewise nonconflicting iff the pairs that comprise it are nonconflicting.

An $L(1,1)$-labeling can be alternately defined as a labeling such that every path in the graph in nonconflicting. Immediately we conceive of a natural relaxation: instead of requiring that every path be nonconflicting, we demand only the existence of at least one nonconflicting path from u to v for every ordered pair $(u, v) \in V(G)^2$ such that u, v are connected. Such a labeling we call a $\widetilde{L}(1,1)$-*labeling*, or *pseudo-schedule*. The corresponding chromatic number is called the *pseudo-(1,1)-number*, denoted $\widetilde{\chi}_{1,1}$.

Finally, we will often find it convenient to work with edges uv such that both (u, v) and (v, u) are nonconflicting (under some labeling); such an edge is said to be *bidirectional*. A subgraph—such as a path or spanning tree—is bidirectional iff all its edges are bidirectional.

Figure 3 gives an example of a pseudo-schedule.

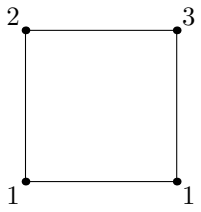


Figure 3: A pseudo-schedule on C_4.

2.1.1 Comparison to other labelings

For the remainder of this section, let G be a graph of order n and degree Δ.

McCormick's seminal 1983 paper [2] proved that

$$\Delta + 1 \leq \chi_{1,1}(G) \leq \Delta^2 + 1.$$

McCormick also establishes an approximation ratio of $O(\sqrt{n})$ for the greedy algorithm and proves that computing the (1,1)-number is NP-hard. The only alternative to the greedy algorithm in the literature is due to Ramanathan and Lloyd [3], who prove an approximation ratio with order of the graph thickness (ie, the minimum number of planar subgraphs into which a graph can be decomposed).

Because $L(1,1)$-labeling is the most common approach to broadcast scheduling, it will be the typical "foil" of the thesis, the standard against which the pseudo-schedule is compared. This chapter will demonstrate that the bounds can be lowered substantially if the pseudo-schedule is an acceptable alternative to strict scheduling, including an $O(\Delta)$ improvement in the upper bound. In §3 we present several pseudo-scheduling algorithms, each with different implementation characteristics, but all sharing an approximation ratio of $O(\Delta)$.

The study of various relaxations and restrictions on colorings is actively pursued by mathematicians and considered interesting in its own right. $L(1)$-labeling (proper coloring) may be relaxed by allowing "defects," in the sense that a coloring is d-defective iff every vertex has at most d adjacent vertices of the same color. Cowen, Goddard, and Jesurum make a review of this line of research in [4].

Pseudo-schedules are in fact defective colorings, as one can see in Figure 3, but not vice-versa; the former have additional structure.

Broersma et al. [5] have introduced the notion of backbone coloring, which is an $L(1)$-labeling with the additional condition that in some specified spanning subgraph, the (integer) colors differ by at least two. By operating with the square of the graph, backbone colorings can be used to generate relaxations of $L(2,1)$-labelings (see [1]) that resemble how pseudo-schedules relax $L(1,1)$-labelings. That said, any backbone coloring of G^2 is an $L(1,1)$-labeling of G, so backbone colorings do not achieve the same level of relaxation as pseudo-schedules.

Suppose that instead of one color, each vertex in G is colored with a set of b colors, with the condition that adjacent vertices must have disjoint color sets. If such a "fractional" coloring can be accomplished using a colors, G is said to be $(a:b)$-colorable. If $f_b(G)$ denotes the smallest a such that G is $(a:b)$-colorable, then since $f_{a+b}(G) \leq f_a(G) + f_b(G)$,

$$\chi_f(G) = \lim_{b\to\infty} \frac{f_b(G)}{b} = \inf_{b\to\infty} \frac{f_b(G)}{b}$$

is well-defined; it is called the fractional chromatic number. (This definition is due to Scheinerman and Ullman [6].) Every coloring is a fractional coloring, so $\chi_f \leq \chi$. That the inequality may be strict is shown by C_5: this graph is $(5:2)$-colorable, so $\chi_f(C_5) \leq \frac{5}{2}$. (In fact, $\chi_f(C_5) = \frac{5}{2}$.) Computing the fractional chromatic number is NP-Hard [7].

Fractional coloring on the graph square is a natural alternative to $L(1,1)$-labeling for broadcast scheduling, although it is only relevant when the medium can be divided into arbitrarily (or at least "very") small segments. Molloy and Reed [8] give an upper bound

$$\chi_f(G) \leq \frac{\omega + \Delta + 1}{2}$$

where ω is the size of the largest clique in G. Operating on G^2, this bound is still $O(\Delta^2)$.

2.2 Relations between the chromatic numbers

Every $L(1,1)$-labeling is a pseudo-schedule, so clearly $\widetilde{\chi}_{1,1} \leq \chi_{1,1}$. It is also obvious that $\chi \leq \chi_{1,1}$. Intuitively one might expect that the pseudo-(1,1)-number lies between the chromatic numbers for $L(1)$- and $L(1,1)$-labeling—but this intuition is badly wrong.

Theorem 2.2. *For any integer k, there exists a graph G such that $\widetilde{\chi}_{1,1}(G) = \chi(G) + k$.*

Proof. If $k \geq 0$, let $G = K_{k+1,1}$ (the star graph with $k+2$ vertices). $\chi(G) = 2$, but it is easy to see that $\widetilde{\chi}_{1,1}(G) = k+2$.

If, on the other hand, $k < 0$, set $G = K_{3-k}$ to begin, and label the vertices $x_1, x_2, \ldots, x_{3-k}$. Now to every x_i attach a new vertex y_i, and to y_i attach a new vertex z_i. Finally, introduce an edge $z_i v_{i+1}$ for all $1 \leq i \leq 2-k$. Figure 4 shows the resulting graph for $k = -1$.

$\chi(G) = 3-k$ due to the $(3-k)$-clique. On the other hand, G admits a pseudo-schedule in three colors: assign all x_i the first color, all y_i the second color, and all z_i the third color. Since two colors is obviously too few, $\widetilde{\chi}_{1,1}(G) = 3$. □

Our intuition misled us because, unlike both $L(1)$- and $L(1,1)$-labeling, or indeed any of the colorings discussed in §2.1.1, the pseudo-schedule is *non-monotonic* with respect to edge insertion; that is, $\widetilde{\chi}_{1,1}$ may either increase or decrease as edges are inserted into a graph. This is due to the fact that the pseudo-schedule relies definitionally on an existence condition, not a universal condition. Thus, for instance, a new edge may permit more color-efficient nonconflicting paths through the graph. For a dramatic example of this, see §2.3.1, particularly Theorem 2.7.

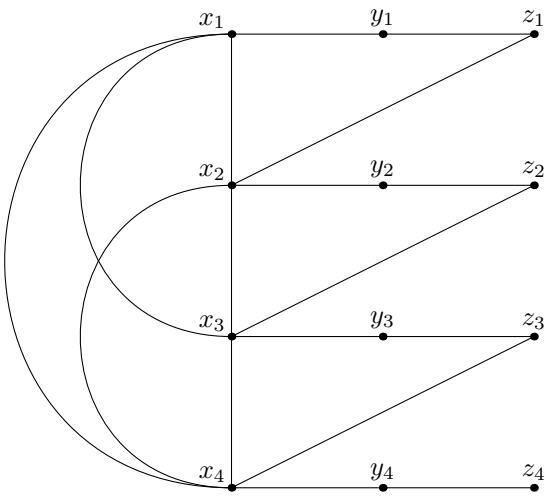


Figure 4: A graph with $\widetilde{\chi}_{1,1} = 3 < \chi = 4$.

2.3 Bounds on $\widetilde{\chi}_{1,1}$

The proof of the bounds is in two parts, corresponding to the lower and upper bound. The lower bound we establish in this section, but the upper bound is a consequence of the analysis of Algorithm 3.3.1 in §3.3.

Lemma 2.3. *Let G be a graph with a pseudo-schedule in k colors; there exists a spanning tree of G with degree less than k.*

Proof. Let s be the pseudo-schedule guaranteed by the hypothesis, and let r be any vertex in G. By definition, s must exhibit some nonconflicting path from any other vertex v to r; by taking the collection of these paths, sans any edges that produce cycles, we obtain a directed spanning tree T of G.

Suppose, for the sake of argument, that T has a vertex z such that $\deg_T(z) \geq k$. It follows that at least two vertices of $N_T[z]$ are monochromatic. Let x be one of these vertices. If $s(x) = s(z)$, neither (x, z) nor (z, x) is nonconflicting; otherwise let $y \neq z$ denote the vertex colored like x in $N_T[z]$. At least one of (x, z) or (y, z) must be an edge in T, but neither is nonconflicting. This contradicts the construction of T, so we conclude that $\Delta_T < k$. □

Corollary 2.4. *Any graph that admits a pseudo-schedule in three colors contains a Hamiltonian path.*

We can now prove the main result of this chapter.

Proof of Theorem 2.1. The lower bound follows from Lemma 2.3, for if there were a pseudo-schedule in Δ^* colors, we could obtain a spanning tree of degree less than Δ^*, which is impossible. The upper bound is established by Theorem 3.5. □

Observe that the upper bound on $\widetilde{\chi}_{1,1}$ is linear in the degree of the graph, whereas it is quadratic for $\chi_{1,1}$. Therefore if the pseudo-schedule constitutes a valid alternative to the $L(1,1)$-labeling, the savings in the number of colors are, in principle, substantial. How this plays out in practice is examined in §3.

The lower bound given in the preceding theorem is tight, as evidenced by any tree. Besides the vacuous example K_2, the only graphs that I know achieve the upper bound are C_{3k+2} $(k \geq 1)$, as we will see in §2.3.1. Hence the upper bound may not in fact be tight; and even if it is, we will see in §2.4 that almost no graph actually hits the bound.

2.3.1 Graphs for which $\widetilde{\chi}_{1,1}$ is known

A pseudo-schedule on any tree T must render every edge of T bidirectional, so $\widetilde{\chi}_{1,1}(T) = \chi_{1,1}(T) = \Delta_T + 1$ [1]. It is also easy to see that for the complete graph K_n, $\widetilde{\chi}_{1,1}(K_n) = n$. We also know the pseudo-(1,1)-numbers for the complete bipartite graphs $K_{m,n}$.

Theorem 2.5. *Let m, n be nonnegative integers such that $m \geq n$. The following statements characterize the pseudo-(1,1)-number of $G = K_{m,n}$:*

- *If $n = 1$ then $\widetilde{\chi}_{1,1}(G) = m + 1$.*
- *If $m = n = 2$ then $\widetilde{\chi}_{1,1}(G) = 3$.*

- *If $m \geq 3$ and $n \geq 2$ then $\widetilde{\chi}_{1,1}(G) = m$.*

Proof. If $n = 1$ then G is a tree. If $m = n = 2$ then $G = C_4$, and we refer to Theorem 2.6 below. For the remaining case, let the vertices of G be labeled $a_1, a_2, \ldots, a_m$ and $b_1, b_2, \ldots, b_n$ as per the partite sets. By definition of pseudo-schedules, for each $1 \leq i \leq m$ there must be some $1 \leq j \leq n$ such that (a_i, b_j) is nonconflicting, which implies that no two a_i may have the same color; hence $\widetilde{\chi}_{1,1}(G) \geq m$.

Now let a_i take color i and b_j take color j; we will prove that this coloring is a pseudo-schedule. Clearly the edge $a_i b_j$ is bidirectional when $i \neq j$. To proceed from a_i to b_i without conflict, we may use any path a_i, b_j, a_k, b_i where i, j, k are distinct. If $m = n$, the same pattern works from b_i to a_i; otherwise simply use b_i, a_m, b_j, a_i with i, j, m distinct. Thus there is a nonconflicting path from any vertex in one partite set to any vertex in the other partite set.

To proceed from a_i to a_m $(i \neq m)$ without conflict, any path a_i, b_j, a_m suffices, provided i, j, m are distinct. From a_i to a_l $(i \neq l)$ where $l < m$, we may use a_i, b_j, a_m, b_k, a_l where i, j, k, l, m are distinct. If $m = n$, the same pattern works from b_i to b_j $(i \neq j)$; otherwise simply use b_i, a_m, b_j with i, j, m distinct. We conclude that there is a nonconflicting path between any two vertices in the same partite set. □

We are also able to characterize the pseudo-(1,1)-number for cycles.

Theorem 2.6. *Let $n \geq 3$.*

$$\widetilde{\chi}_{1,1}(C_n) = \begin{cases} 3, & \text{if } n \bmod 3 \leq 1 \\ 4, & \text{otherwise} \end{cases}.$$

Proof. Label the vertices $v_1, v_2, \ldots, v_n$ along the cycle. From Theorem 2.1 we know that a cycle's pseudo-schedule requires at least three colors. If $n \bmod 3 \leq 1$, let v_i take color $i \bmod 3$; then the path $v_1, v_2, \ldots, v_n$ is bidirectional.

In the remaining case $n \bmod 3 = 2$, suppose C_n admits a pseudo-schedule s in three colors. Consider the vertices v_1, v_2, v_3, v_4: at least two must share a color. These cannot be v_i and v_{i+2}, for then there would be no nonconflicting path into v_{i+1}. Suppose, then, that $s(v_2) = s(v_3)$. Relabel the vertices by decrementing every index, but with v_1 relabeled as v_n, and consider afresh the vertices $v_1, \ldots, v_4$. If $s(v_2) = s(v_3)$ again, there is no nonconflicting path from v_2 to v_3 since $s(v_1) = s(v_n)$—we are "blocked" on both sides. Hence it must be that $s(v_1) = s(v_4)$.

Now consider v_4, v_5, v_6, v_7. By similar arguments as above, we find $s(v_4) = s(v_7)$. Continuing in this fashion, we have $s(v_1) = s(v_4) = \cdots = s(v_{n-1})$. But then there is no nonconflicting path into v_n, so we conclude that $\widetilde{\chi}_{1,1}(C_n) \geq 4$.

We can readily construct a pseudo-schedule on C_n in four colors: for $1 \leq i \leq n-2$, assign v_i the color $i \bmod 3$; assign v_{n-1} the color 3; and assign v_n the color 1. It is easy to confirm that the path $v_1, v_2, \ldots, v_n$ is bidirectional. □

We conclude this section with an exploration of the *wheel graph* W_n, which is C_{n-1} plus an additional vertex adjacent to all others.

Theorem 2.7. *Let $n \geq 4$.*

$$\widetilde{\chi}_{1,1}(W_n) = \begin{cases} 4, & \text{if } n \leq 5 \\ 5, & \text{otherwise} \end{cases}.$$

Proof. Label the vertices along the circumference $v_1, v_2, \ldots, v_{n-1}$ and let v_n be the dominating vertex. Under any pseudo-schedule s, there must exist some $1 \leq i \leq n-1$ such that (v_i, v_n) is nonconflicting. Hence $s(v_i)$ is unique in s, which forces s to use at least four colors. This obviously suffices for W_4; for W_5, color the major cycle as in the proof of Theorem 2.6 and assign v_n the fourth color.

It remains to treat W_n for $n \geq 6$. Consider the dominating vertex v_n: if $s(v_n) = s(v_i)$ for any $1 \leq i \leq n-1$, note that there is no nonconflicting path out of v_i. We conclude that $s(v_n)$ is unique in s, which forces s to use at least five

colors. Color, preliminarily, the major cycle as in the proof of Theorem 2.6, and give the fifth color to v_n. If $(n-1) \bmod 3 = 2$, there is a bidirectional spanning tree of W_n comprised of the path $v_1, v_2, \ldots, v_{n-1}$ plus the edge $v_{n-2}v_n$. If, alternately, $(n-1) \bmod 3 \neq 2$, recolor v_{n-1} with the fourth color; the path $v_1, v_2, \ldots, v_{n-1}$ and the edge $v_{n-1}v_n$ are bidirectional. □

W_n provides a dramatic example of the non-monotonicity of the pseudo-schedule. Note that $K_{n-1,1}$ requires n colors, yet it is a subgraph of W_n, which requires at most five colors, regardless of how large n may be.

2.4 Bounds on $\widetilde{\chi}_{1,1}$ for almost every graph

A *random graph model* $G(n,p)$ is the probability space of graphs with n vertices such that each of the $\binom{n}{2}$ possible edges is chosen independently with probability $0 \leq p \leq 1$, where p may be an arbitrary function, although typically $p = p(n)$. A *random graph* is a graph selected from $G(n,p)$; as shorthand we say "a (random) graph $G = G(n,p)$." Finally, a statement P_n is said to hold *almost always* iff $\Pr[P_n] \to 1$ as $n \to \infty$; if the statement P_n concerns the random graphs $G = G(n,p)$, we say it holds for *almost every (random) graph*.

Our task in this section will be to improve the upper bound on $\widetilde{\chi}_{1,1}$ by restricting our attention to "common" graphs, thereby avoiding the structural "oddities" that force up the universal upper bound. The definitive modern treatise on random graphs is due to Bollobás [9], and the results of this section rely critically on theorems presented in that work.

Theorem 2.1 shows that $\widetilde{\chi}_{1,1}(G) \leq 2\Delta_G$. As it turns out, the inequality is almost always strict under very broad conditions.

Theorem 2.8. *Let $p = p(n)$ satisfy $pn/\log n \to \infty$; then almost every random graph $G = G(n,p)$ has $\widetilde{\chi}_{1,1}(G) < 2\Delta_G$.*

Proof. Suppose $p \leq \frac{1}{2}$. Theorem 3.9 of [9] states that almost every G has a unique vertex r of maximum degree. It suffices to consider only the component containing r, so assume without loss of generality that G is connected. Now consider the operation of Algorithm 3.3.1 (p. 30) when r is specified as the root. The algorithm will create a spanning tree T of G and examine, for each vertex v, the neighborhood of the T-parent of v as well as the T-parents of (some) vertices adjacent to v. Including v, the number of vertices examined is no more than $2\Delta_G - 1$, so it is not possible for the algorithm to use more than this number of colors.

Suppose $p > \frac{1}{2}$. Riordan and Selby [10] show that almost every G has $\Delta_G > n/2$, in which case we obviously require fewer than $2\Delta_G$ colors. □

The preceding theorem is somewhat interesting, but what one really wants is (at least) a reduction of the multiplicative factor of Δ. The following theorem shows this is possible if p is held constant. We will utilize some standard asymptotic notation: for any function $g(n)$, let $o(g)$ denote the class of functions $f(n)$ such that $f/g \to 0$ as $n \to \infty$; we also use $o(g)$ to denote an arbitrary member of the class. Related to this, we say $f \sim g \iff f(n) = (1 + o(1))g$.

Theorem 2.9. *Let $0 < p < 1$ be fixed. If $p \leq \frac{1}{2}$ then*

$$\frac{\widetilde{\chi}_{1,1}(G)}{\Delta_G} < 1 + 2(\sqrt{2} - 1)(1 - p) + o(1)$$

for almost every random graph $G = G(n, p)$. If $p > \frac{1}{2}$ then

$$\frac{\widetilde{\chi}_{1,1}(G)}{\Delta_G} < 2 - p + o(1)$$

for almost every random graph $G = G(n, p)$.

Proof. Corollary 10.11 of [9] tells us that almost every G has diameter two. Let $\Delta = \Delta_G$ and r be a vertex of maximum degree; once again we invoke Algorithm 3.3.1 with root r, and let T be the spanning tree constructed by the algorithm. Obviously T has height (at most) two, and for this reason, the number

of colors required beyond $\Delta + 1$ is determined entirely by the vertices at distance two from r. (If there are none, clearly $\widetilde{\chi}_{1,1}(G) \leq \Delta + 1$.) In particular, if such a vertex v has T-parent u, we need examine at most $\deg(u)+\deg(v)$ different vertices (including v). However, a vertex adjacent to v need only be counted if it is also at distance two from r, since the T-parent of a vertex at distance one is r itself, which is already examined in the neighborhood of u. Hence

$$\widetilde{\chi}_{1,1}(G) \leq \deg(u) + \alpha + 1 \leq \Delta + \alpha + 1$$

where α is the maximum number of vertices at distance two from r adjacent to any single vertex also at distance two from r.

Observe that if Δ is large, α should be small since "lots" of vertices will be adjacent to r. Alternately, if Δ is small, this implies p is small, and α should be small because high-degree vertices are unlikely. Intuitively, then, we hope that α should be substantially smaller than Δ almost always.

Instead of attempting to calculate α directly, let us examine the simpler random variable X_v that counts the number of r-distance-two neighbors of a given r-distance-two vertex v. There are Δ vertices adjacent to r, one of which is the T-parent of v, leaving $\Delta - 1$ that may be selected at random. There are $n - \Delta - 1$ vertices at distance two from r, one of which is v itself, so $X_v \leq n - \Delta - 2$. We can imagine that v selects neighbors by the following process: first $\Delta - 1$ potential neighbors are chosen from between the two groups, and each potential neighbor is connected with probability p. X_v then counts the actual number of r-distance-two neighbors.

The probability distribution $\Pr[X_v = k]$ is known as Fisher's noncentral hypergeometric distribution. Let μ be the expectation of X_v; an asymptotic approx-

imation for μ is given by Levin [11]. In particular,

$$\begin{aligned}\mu &\to \frac{2(n-\Delta-2)(\Delta-1)p}{(n-3)p+\sqrt{(n-3)^2p^2+4(1-p)(n-\Delta-2)(\Delta-1)p}} \\ &\to \frac{2(n-\Delta)\Delta p}{np+\sqrt{n^2p^2+4(1-p)(n-\Delta)\Delta p}} \\ &< \frac{2(n-\Delta)\Delta p}{np+\sqrt{n^2p^2}} \\ &= \Delta - \frac{\Delta^2}{n}\end{aligned}$$

where elimination of additive constants is justified by the fact that both $n, \Delta \to \infty$, the latter being a consequence of $\Delta \sim np$ almost always (Corollary 3.14 of [9]). To continue:

$$\lim_{n\to\infty} \mu < \left(1 - \frac{\Delta}{n}\right)\Delta \to (1-p)\Delta.$$

The variance σ^2 of X_v has

$$\sigma^2 \to \left(\frac{2}{\mu} + \frac{1}{n-\Delta-\mu-2} + \frac{1}{\Delta-\mu-1}\right)^{-1}$$

according to [11]. So in fact

$$\frac{\sigma^2}{\mu^2} \to \left(2\mu + \frac{\mu^2}{n-\Delta-\mu-2} + \frac{\mu^2}{\Delta-\mu-1}\right)^{-1} \to 0$$

and thus per Chebyshev's inequality, $X_v \sim \mu$ almost always.

The random variable α is the maximum of the set of random variables $\{X_v; \mathrm{dist}_G(r,v) = 2\}$. The random variables in this set are not independent, but they are "almost independent" in the sense of Dysktra et al. [12], so we can treat them as if they were independent for large n. But since any X_v is almost always close to μ, so must be α; that is to say, $\alpha \sim \mu$ almost always. Thus

$$\widetilde{\chi}_{1,1}(G) \leq \Delta + (1+o(1))\mu + 1 < \Delta + (1+o(1))(1-p)\Delta + 1$$

almost always. Dividing through by Δ yields the desired result.

This result is fine for $p > \frac{1}{2}$, but it fails to capture our previous intuition that α ought to be small when p is small. Assuming $p \leq \frac{1}{2}$, let us reexamine the square root term:

$$\begin{aligned}\sqrt{n^2p^2 + 4(1-p)(n-\Delta)\Delta p} &> \sqrt{n^2p^2 + 4(1-p)(1-\Delta/n)n^2p^2} \\ &= np\sqrt{1 + 4(1-p)(1-\Delta/n)} \\ &\to np\sqrt{1 + 4(1-p)^2} \\ &\geq \sqrt{2}np.\end{aligned}$$

Here we have used the fact that $\Delta > np$ almost always [10] and that $\Delta \sim np$ almost always. Finally we obtain

$$\begin{aligned}\lim_{n\to\infty} \mu &< \frac{2(n-\Delta)\Delta p}{(1+\sqrt{2})np} \\ &\to 2(\sqrt{2}-1)(1-p)\Delta\end{aligned}$$

from which derives the desired result for $p < \frac{1}{2}$. □

The theorem yields some nice corollaries.

Corollary 2.10. *Almost every graph $G = G(n, \frac{1}{2})$ has*

$$\frac{\widetilde{\chi}_{1,1}(G)}{\Delta_G} < \sqrt{2} + o(1).$$

Interestingly, we are able to say something meaningful about almost every random graph $G = G(n, p)$ with constant p, even when we do not know the value of p.

Corollary 2.11. *Let $0 < p < 1$ be fixed. Almost every graph $G = G(n, p)$ has*

$$\frac{\widetilde{\chi}_{1,1}(G)}{\Delta_G} < (2\sqrt{2} - 1) + o(1).$$

We note that the proof of the $p > \frac{1}{2}$ case actually goes through for a rather wide range of p.

Theorem 2.12. *Let $p = p(n)$ satisfy $p^2 n - 2\log n \to \infty$ and $n^2(1-p) \to \infty$. For almost every random graph $G = G(n,p)$,*

$$\frac{\widetilde{\chi}_{1,1}(G)}{\Delta_G} < 2 - p + o(1).$$

Before leaving this section, let us note that the methods used in the preceding proofs are relatively crude, and the results are quite unlikely to be sharp; indeed, I suspect that the "true" multiplicative factor of Δ in the upper bound for almost all graphs is unity. It is even possible that the upper bound for almost all graphs is properly seen as a function of Δ^*. We will return to these issues when we conclude the thesis in §6.

List of References

[1] T. Calamoneri, "The $L(h,k)$-labelling problem: A survey and annotated bibliography," *The Computer Journal*, vol. 49, no. 5, pp. 585–606, 2006.

[2] S. T. McCormick, "Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem," *Mathematical Programming*, vol. 26, pp. 153–171, 1983.

[3] S. Ramanathan and E. L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 2, pp. 166–177, 1993.

[4] L. Cowen, W. Goddard, and C. E. Jesurum, "Defective coloring revisited," *Journal of Graph Theory*, vol. 24, no. 3, pp. 205–219, 1997.

[5] H. Broersma, F. V. Fomin, P. A. Golovach, and G. J. Woeginger, "Backbone colorings for graphs: tree and path backbones," *Journal of Graph Theory*, vol. 55, no. 2, pp. 137–152, 2007.

[6] E. R. Scheinerman and D. H. Ullman, *Fractional Graph Theory*. Self-published, 2008. [Online]. Available: http://www.ams.jhu.edu/~ers/fgt

[7] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1980.

[8] M. Molloy and B. Reed, *Graph Colouring and the Probabilistic Method*, ser. Algorithms and Combinatorics. Springer, 2002.

[9] B. Bollobás, *Random Graphs*, 2nd ed., ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

[10] O. Riordan and A. Selby, "The maximum degree of a random graph," *Combinatorics, Probability and Computing*, vol. 9, pp. 549–572, 2000.

[11] B. Levin, "Simple improvements on Cornfield's approximation to the mean of a noncentral hypergeometric random variable," *Biometrika*, vol. 71, no. 3, pp. 630–632, 1984.

[12] R. L. Dysktra, J. E. Hewett, and W. A. Thompson, Jr., "Events which are almost independent," *The Annals of Statistics*, vol. 1, no. 3, pp. 674–681, 1973.

CHAPTER 3

Algorithms

In the previous chapter we investigated the fundamental nature of the pseudo-schedule and its associated chromatic number, but even there we found it useful to invoke algorithmic arguments, or even particular algorithms. The current chapter adopts the algorithmic point of view entirely, offering several alternative pseudo-scheduling procedures and evaluating their trade-offs. The fundamental performance metric will be the number of colors used, although we will permit worse performance on this axis in exchange for other desirable properties. Our findings are summed up broadly in Table 1.

Before the discussing the algorithms in earnest, we will describe a special type of pseudo-schedule that, in some sense, embeds a strictly-scheduled spanning tree; all the algorithms discussed in this section will use this special pseudo-schedule. Finally, before concluding, we will examine a customized algorithm for pseudo-scheduling dismantleable graphs.

3.1 Spanning tree pseudo-schedules

Given a graph G with spanning subgraph H, we say that a pseudo-schedule s is an *H-pseudo-schedule* iff every edge of H is bidirectional in G relative to s.

Algorithm	Worst Case Colors (Dominant Term)	Avg Case Colors (Rule of Thumb)	As Protocol Candidate
Greedy (§3.2)	Δ^2	near Δ	Poor
2Δ (§3.3)	2Δ	Δ to 2Δ	Mediocre
d-band (§3.4)	$2d\Delta$	near $d\Delta$	Good
3-band	6Δ	near 3Δ	Very Good

Table 1: Summary comparison of the main algorithms of this chapter. We list the $d = 3$ instantiation of d-band separately because of its particular usefulness.

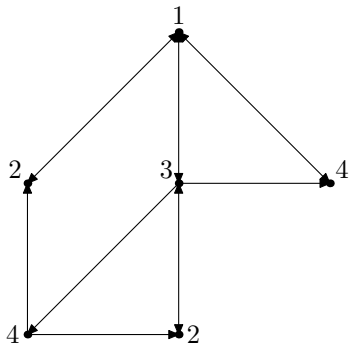


Figure 5: A pseudo-schedule that is not spanning tree; arrows indicate nonconflict.

We are particularly interested in the case where H is a spanning tree, as this corresponds to the concept of a routing tree in a network. Thus we say that a pseudo-schedule s of G is a *spanning tree pseudo-schedule* or *st-pseudo-schedule* iff there exists some spanning tree T of G such that s is a T-pseudo-schedule. Not every pseudo-schedule is spanning tree, as Figure 5 demonstrates.

The chromatic number associated with the st-pseudo-schedule is the *st-pseudo-(1,1)-number*, denoted $\widehat{\chi}_{1,1}$. Clearly $\widetilde{\chi}_{1,1} \leq \widehat{\chi}_{1,1}$, but I do not know if the inequality is ever strict. (The reader can confirm that $\widetilde{\chi}_{1,1} = \widehat{\chi}_{1,1} = 4$ in the graph of Figure 5.) Indeed, we can replace $\widetilde{\chi}_{1,1}$ with $\widehat{\chi}_{1,1}$ in all of the results of the previous chapter, which notably implies that all the graphs in §2.3.1 have $\widetilde{\chi}_{1,1} = \widehat{\chi}_{1,1}$.

As a step towards resolving the question of the equality, let us observe that it holds in the smallest nontrivial case.

Theorem 3.1. *A pseudo-schedule in three colors is spanning tree.*

Proof. Let s be a pseudo-schedule on G in three colors. Suppose, for the sake of argument, that s is not spanning tree; then there must be a cyclic nonconflicting path $P = x, y, \ldots, z, x$ containing at least one edge that is not bidirectional. Without loss of generality, let xz be this edge, with (z, x) nonconflicting but not (x, z).

Now because (x, z) is conflicting, there is some vertex w adjacent to z such

that $s(x) = s(w)$. There must be some nonconflicting path P' from w to x. Let (p', p) be the first nonconflicting pair from P' such that $p \in P$; in other words, p' is the vertex immediately before P' "hits" P for the first time (possibly $p' = w$ or $p = x$). Let p be preceded in P by o and followed by q; without loss of generality, assume $s(o) = 1, s(p) = 2, s(q) = 3$. Hence $s(p') \notin \{1, 2, 3\}$—a contradiction. We conclude that s is spanning tree. □

Corollary 3.2. *For any graph G, $\widetilde{\chi}_{1,1}(G) = 3 \iff \widehat{\chi}_{1,1}(G) = 3$.*

3.2 The greedy algorithm

Let G be a graph with spanning tree T and vertices $v_1, v_2, \ldots, v_n$. We can produce a pseudo-schedule on G as follows:

1. Set $i = 1$.
2. Color v_i with the lowest positive integer k such that for all $x \in N_T[v_i]$, k does not appear in $N_G[x]$, and for all $y \in N_G[v_i] - N_T[v_i]$, k does not appear in $N_T[y] - \{y\}$. (In other words, k does not appear on any T-neighbor of v, any G-neighbor of any T-neighbor of v, or any T-neighbor of any G-neighbor of v.)
3. Increment i and go to step 2.

This algorithm was first presented by Ren in §2.4.2 of his doctoral thesis [1], but we are able to take the analysis of the algorithm much further here.

3.2.1 Proof of correctness

We shall prove that the coloring s produced by the greedy algorithm is a T-pseudo-schedule. Let $uv \in E(T)$. Clearly $s(u) \neq s(v)$. Let w be a G-neighbor of v, and observe that w is thereby a G-neighbor of a T-neighbor of u; or equivalently, u is a T-neighbor of a G-neighbor of w. Thus $s(u) \neq s(w)$ regardless of whether u

or w is colored first, so (u, v) is a nonconflicting pair. By an identical argument, (v, u) is nonconflicting; hence uv is bidirectional.

3.2.2 Analysis

Theorem 3.3. *Let h be the number of colors used by the greedy algorithm on graph G with spanning tree T. Then*

$$\Delta_T + 1 \leq h \leq 2\Delta_G\Delta_T - (\Delta_T)^2 + 1.$$

Furthermore, these bounds are tight.

Proof. The lower bound is simply the $(1, 1)$-number of T [2]. To obtain the upper bound, we count the number of vertices examined by the algorithm before coloring a vertex v. Every T-neighbor of v must be colored differently than v, as must their neighborhoods: at most $\Delta_G\Delta_T$ distinct vertices. For the remaining $\Delta_G - \Delta_T$ neighbors of v, v need only differ from their T-neighbors, giving at most $(\Delta_G - \Delta_T)\Delta_T$ distinct vertices. Adding the totals plus one (for v itself) gives the upper bound.

The tightness of the lower bound is obvious. The tightness of the upper bound is established by degenerate case. Fix some positive integer $m \geq 2$ and let R_i be a full $(m-1)$-ary tree with height $2i-1$ where $1 \leq i \leq m-1$. Consider any R_i with the natural root r_i: we claim that r_i and its neighbors can be colored by the algorithm in such a way that colors $(i-1)m+1, \ldots, im$ appear on them, in whatever way we like. This is easy to see for $R_1 = K_{m,1}$, and it follows for the remaining R_i by induction.

By R_{m-1} alone we can induce $m^2 - m$ colors, but the final $m+1$ additional colors require more subtlety. Let Q be a tree with root q connected to vertices $x_1, \ldots, x_{m-1}$. For each x_j, let $R_{j,k}$ be a copy of R_k $(1 \leq k \leq m-1)$ with its natural root connected to x_j. Finally, construct G by introducing a vertex v connected to

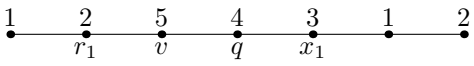


Figure 6: Degree-2 degenerate case for the greedy algorithm.

the roots $r_1, \ldots, r_{m-1}$ and q. Note that G is a tree with degree m.

G can now be colored by the greedy algorithm such that R_i exhibits colors $(i-1)m+1, \ldots, im$ in the (closed) neighborhood of its root r_i, with r_i itself receiving color im. In Q, we arrange for the root of $R_{j,k}$ to take the color $(k-1)m+j$, with the remaining colors appearing in the neighborhood of the root. After this, x_1 receives color $m^2 - m + 1$, as it is within distance two of all preceding colors. Similarly, x_2 gets color $m^2 - m + 2$, as it is within distance two of all preceding colors as well as x_1. Running through the x_j gets the number of colors to $m^2 - 1$.

We now come to q and note that it is, via Q, within distance two of all colors up to $m^2 - 1$ except im for $1 \leq i \leq m - 1$. But these are precisely the colors appearing at the r_i, so q is colored m^2. Finally, v is colored $m^2 + 1$.

The degenerate case for $m = 2$ is given in Figure 6. □

Note that the worst-case performance of the greedy algorithm is no worse than the worst-case performance of the greedy algorithm for strict scheduling, which is $\Delta_G^2 + 1$ colors [3]; indeed, the worst case is strictly superior if $\Delta_T < \Delta_G$.

The *approximation ratio* of an algorithm is the ratio of its cost to the optimal cost. We are able to demonstrate that the performance ratio of the greedy algorithm is $O(\Delta_G)$ if one uses the spanning tree of Fürer and Raghavachari [4], which is guaranteed to have degree at most one more than the smallest possible, and which can be determined in polynomial time.

Theorem 3.4. *Given a graph with degree Δ, the approximation ratio of the greedy algorithm is less than 2Δ, provided the Fürer-Raghavachari spanning tree is used.*

Proof. Let h denote the number of colors used by the algorithm on G, and k denote

the optimal number. If we let Δ^* denote the minimum degree of a spanning tree, we have

$$\frac{h}{k} \leq \frac{h}{\Delta^* + 1}$$

by Theorem 2.1.

Fürer and Raghavachari [4] give a polynomial-time algorithm for finding a spanning tree with degree at most $\Delta^* + 1$; using this tree yields

$$\begin{aligned}\frac{h}{k} &\leq \frac{2\Delta(\Delta^* + 1) - (\Delta^* + 1)^2 + 1}{\Delta^* + 1} \\ &= 2\Delta - (\Delta^* + 1) + \frac{1}{\Delta^* + 1} \\ &< 2\Delta.\end{aligned}$$

□

If we are unable to say anything about the ratio of Δ_T to Δ^*, the approximation ratio blows up to $O(\Delta_G \Delta_T) = O(\Delta_G^2)$.

3.2.3 Implementation issues

Under the name RAC-CT, Ren [1] implemented the greedy algorithm as a MAC protocol for broadcast networks. The greedy algorithm has some qualities to recommend it as a protocol candidate, first and foremost its conceptual simplicity. Empirically it also seems to do much better than the worst case: Ren reports performance close to Δ for random grids, and Table 2 suggests that the algorithm performs well below Δ on $G(n, p)$ with constant p. The user is also granted a great degree of freedom since he chooses both the spanning tree and the coloring order; the choice of the spanning tree is particularly important in network applications, since this corresponds to the choice of routing tree.

On the other hand, several factors militate against the greedy algorithm as a protocol candidate. However uncommon the worst case might be in practice,

n	$p = 0.1$	$p = 0.25$	$p = 0.5$
10	1.024	0.916	0.498
20	0.946	0.324	-0.8
30	0.796	-0.588	-1.694
40	0.632	-1.292	-3.012
50	0.46	-1.674	-3.562
60	0.064	-2.83	-4.146
70	-0.41	-2.978	-5.698
80	-0.528	-4.002	-6.144
90	-0.616	-4.82	-5.912
100	-1.08	-5.224	-7.608

Table 2: Empirical performance of the greedy algorithm on random graphs. Rows give the number of vertices and columns the probability of edge existence. For each graph instance we take the average number of colors used in five runs with different coloring orders on an arbitrary BFS spanning tree; then subtract the graph degree to obtain the instance performance. A hundred instances were generated for each configuration; the table reports the average of the performances.

it still has to be accommodated, increasing message size and/or complexity. In a similar vein, observe that the algorithm extensively uses information at distance two from the vertex it is currently coloring. In networks it is generally not possible to obtain such information except by way of an adjacent node.

Perhaps most damagingly, because a vertex cannot be colored while another vertex it must check is being colored, the greedy algorithm requires either central control, token-passing, or mutual exclusion on the two-hop neighborhood. Although Rhee et al. provide a distributed implementation of mutual exclusion in their work on DRAND [5], which Ren used as a basis for RAC-CT, this is expensive in time and bandwidth and scales poorly.

3.3 The 2Δ algorithm

The high upper bound for the greedy algorithm is due to the freedom afforded the user in specifying how the algorithm operates. If we restrict—or indeed, almost eliminate—this freedom, the upper bound can be dramatically improved.

This is the idea behind the 2Δ algorithm for pseudo-scheduling, presented as Algorithm 3.3.1.

Some notation: for any tree T rooted at r, we say that u is the *parent* of v iff $uv \in E(T)$ and $\mathrm{dist}_T(u,r) < \mathrm{dist}_T(v,r)$; we write $u = \mathrm{par}_{T,r}(v)$. (It will be convenient to let $\mathrm{par}_{T,r}(r) = r$.) Naturally we say that v is a *child* of u. We also henceforth permit ourselves the following natural abuse of notation: given a coloring s, which may be only partially defined, for any set U of vertices let $s(U)$ denote the set $\{s(u); u \in U, s(u) \text{ is defined}\}$.

Algorithm 3.3.1: The 2Δ algorithm for pseudo-scheduling

Input: G, a graph; and r, a distinguished vertex of G.
Output: An st-pseudo-schedule on G.

$V_T \leftarrow \{r\}, E_T \leftarrow \emptyset$
$T \leftarrow (V_T, E_T)$
$Q \leftarrow \{r\}, Q' \leftarrow \emptyset$ `// Q, Q' are queues`
$s \leftarrow \emptyset$
repeat
  foreach $v \in Q$ *(in order)* **do**
    $N \leftarrow N_G[v] - V_T$
    append N to Q' (in any order)
    $V_T \leftarrow V_T \cup N$
    $E_T \leftarrow E_T \cup \{vx; x \in N\}$
    $K \leftarrow s(N_G[\mathrm{par}_{T,r}(v)])$
    foreach $x \in N_G[v]$ **do**
      if $x \neq v$ *and* $vx \notin E_T$ **then** add $s(\mathrm{par}_{T,r}(x))$ to K
    $k \leftarrow \min(\mathbb{Z}^+ - K)$
    add $v \mapsto k$ to s
  $Q \leftarrow Q', Q' \leftarrow \emptyset$
until $Q = \emptyset$
return s

3.3.1 Proof of correctness

Let us prove that the coloring s produced by the algorithm given graph G and distinguished vertex r is a T-pseudo-schedule, where T is the tree generated internally by the algorithm. Observe that T is produced by a breadth-first search

process and that every vertex at r-distance i is colored before any vertex at r-distance $i+1$. We can also see that if vertices u, v have distinct parents p_u, p_v respectively, then if p_u was colored before p_v, u was colored before v.

Take $uv \in E(T)$ such that $u = \mathrm{par}_{T,r}(v)$. First we show that (u, v) is nonconflicting. $s(u) \neq s(v)$ clearly. Consider next any child x of v; since u is adjacent to $v = \mathrm{par}_{T,r}(x)$, we have $s(u) \neq s(x)$. The only vertices left to check are those in $N_G[v] - N_T[v]$; let y be such a vertex. Now if $\mathrm{dist}_G(r, u) < \mathrm{dist}_G(r, y)$, then y was colored after u, so $s(u) \neq s(y)$. If, on the other hand, $\mathrm{dist}_G(r, u) = \mathrm{dist}_G(r, y)$, then u "adopted" v before y could (line 10), which implies that u was colored before y, hence $s(y) \neq s(x)$.

Let us now establish that (v, u) is nonconflicting. Obviously $s(v) \neq s(x)$ for any $x \in N_G[u]$ with $\mathrm{dist}_G(r, x) < \mathrm{dist}_G(r, v)$ since in this case x must have been colored before v. Turning to $x \in N_G[u]$ with $\mathrm{dist}_G(r, x) = \mathrm{dist}_G(r, v)$, clearly $s(v) \neq s(x)$ if x is a child of u, since v will be checked before coloring x and vice-versa. If, on the other hand, $\mathrm{par}_{T,r}(x) = w \neq u$, it must be that w was colored before u since the former adopted x, thus $s(x)$ was defined when the algorithm computed $s(N_G[u])$ (line 11) before coloring v, and $s(v) \neq s(x)$.

3.3.2 Analysis

The next theorem indicates that the algorithm was not named capriciously.

Theorem 3.5. *Let h be the number of colors used by Algorithm 3.3.1 on a nontrivial graph G with distinguished vertex r. Then*

$$\deg_G(r) + 1 \leq h \leq 2\Delta_G.$$

Furthermore, these bounds are tight.

Proof. The lower bound and its tightness are obvious. To establish the upper bound, note that in order to color a vertex v, the algorithm must check the parent

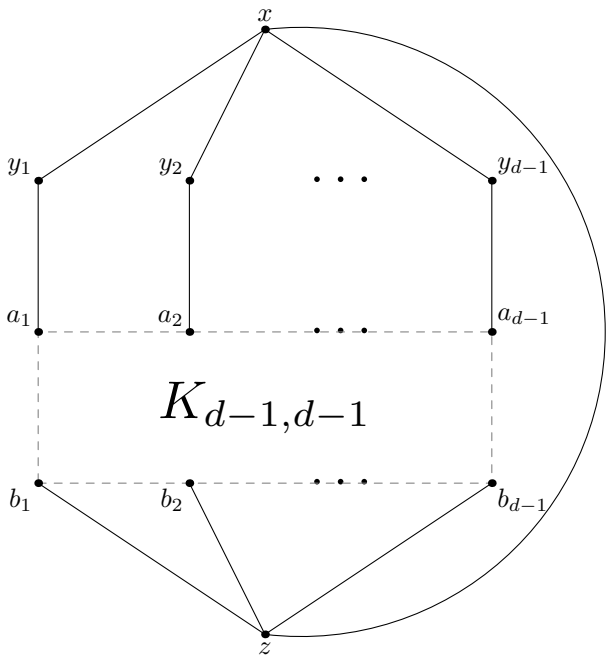


Figure 7: A graph for which the 2Δ algorithm hits the upper bound: J_d,described in the proof of Theorem 3.5, with x as the distinguished vertex.

u of v, every vertex adjacent to u, and additionally the parents of any vertex adjacent to v in G but not T, for a total of at most $2\Delta_G$ vertices (including v).

To establish that the upper bound is tight, we introduce the graph J_d where $d \geq 2$. To construct J_d, begin with the complete bipartite graph $K_{d-1,d-1}$ with partite sets A, B. For each vertex a_i of A, introduce a new vertex y_i adjacent to a_i, and then introduce a vertex x adjacent to all y_i $(1 \leq i \leq d-1)$. Finally, introduce a vertex z adjacent to every vertex in B, and set x, z adjacent. Figure 7 shows the structure of J_d. Letting $r = x$, it is not difficult to see that the algorithm hits the upper bound. (Note that $J_2 = C_5$.) □

Finally, the approximation ratio is

$$\frac{2\Delta}{\Delta^* + 1} \leq \frac{2}{3}\Delta$$

for any graph with degree Δ and a minimum degree spanning tree of degree Δ^*.

3.3.3 Implementation issues

The 2Δ algorithm is primarily distinguished by its excellent worst-case performance, but it sacrifices much in order to achieve it, in particular the user's freedom to choose the spanning tree, although it should be noted that the ability to select the distinguished vertex is modestly helpful, since this vertex can be chosen to correspond to a source/sink/central node in a network. Conceptually the algorithm is more complex than the greedy algorithm, but not by much.

Reliance on information at distance two is reduced significantly relative to the greedy algorithm, but remains substantial. The 2Δ algorithm also requires either central control, token-passing, or mutual exclusion in the two-hop neighborhood. In general, this algorithm is a weak protocol candidate, but very useful for mathematical investigations, as we saw in §2.

3.4 The d-band algorithm

Let G be a graph with a spanning tree T rooted at r. The d-band algorithm on this input produces a T-pseudo-schedule on G, provided that d is sufficiently large (see §3.4.1). Intuitively, the algorithm divides the graph into d bands based on vertices' T-distance from r modulo d, with each band being colored from its own palette, disjoint from every other. The idea is that we can thereby rule out many conflicts *a priori*, greatly reducing the number of vertices that have to be checked.

For the remainder of this section, we implicitly take all terms with respect to G, T, r unless otherwise noted.

The d-band algorithm is distributed and decentralized, with each vertex acting as an autonomous agent passing the following messages:

- REQ-COL(L), where L is a set of excluded colors;

- PUT-COL(x,k), where x is a vertex being assigned color k;
- RPT-COL(k,w), where k is the sender's color (if known) and w is a vertex that must be colored before the sender is colored; or, in a "reverse report," where k is a color excluded for the sender;
- RPT-PAR(k,w), where k is the color of the sender's parent (if known) and w is a vertex that must be colored before the sender's parent is colored; or, in a "reverse report," where k is the color of w, a stepparent of the sender;
- DEP-REQ(w), where w is a vertex whose color must be assigned before the sender can issue REQ-COL; and
- DEP-PUT(w), where w is a vertex whose color must be assigned before the sender can issue PUT-COL.

The sending vertex is implicitly included in any message, along with information about the intended receiver. We let ∞ denote an unknown color and let

$$\text{Palette}(v) = \{\text{dist}_T(r, v) \bmod d + id; i \geq 0\}.$$

Along with the definitions of parent and child as in §3.3, we say also that u is a *stepparent* of v iff $\text{dist}_T(u, r) = \text{dist}_T(v, r) - 1$ and $uv \in E(G) - E(T)$; and that x is a *stepchild* of y iff $\text{dist}_T(x, r) = \text{dist}_T(y, r) + 1$ and $xy \in E(G) - E(T)$. Each vertex is assumed to know its parent, children, stepparents, and stepchildren. Additionally, each vertex knows its T-distance from r. Finally, we assume that $V(G)$ admits a strict total order $\prec$ that can be efficiently computed at any vertex.[1]

The root vertex r assigns itself the color 0, making this known by sending RPT-COL(0,r) to its children. Any vertex besides r must acquire its color as per

[1]The order $\prec$ does not in fact need to be total, but to state the "tight" requirement would contribute more to verbiage than understanding.

AcquireColor (Algorithm 3.5.2, p. 49). Any vertex with children must assign colors to its children as per AssignColors (Algorithm 3.5.3, p. 50). Finally, any non-root vertex must receive and relay reports as per ReportColors (Algorithm 3.5.4, p. 51). The ensemble of these procedures, running independently and in parallel on every vertex simultaneously, constitutes the d-band algorithm.

We assume fully reliable transmission with synchronous communication primitives send and listen. AssignColors uses the primitive "ack-send $\mathcal{M}$ to x" by which is meant: send $\mathcal{M}$ to x and wait until $\mathcal{M}$ is sent back as confirmation, queuing any messages that arrive in the meanwhile for retrieval by the next call to listen.

The ensemble action of the algorithm about a vertex v can be sketched roughly as follows:

1. v listens for RPT-PARs from all of its stepchildren, building a list of excluded colors L.

2. v sends REQ-COL(L) to its parent u.

3. u listens for RPT-COLs from all of its stepchildren, building a list of forbidden colors K.

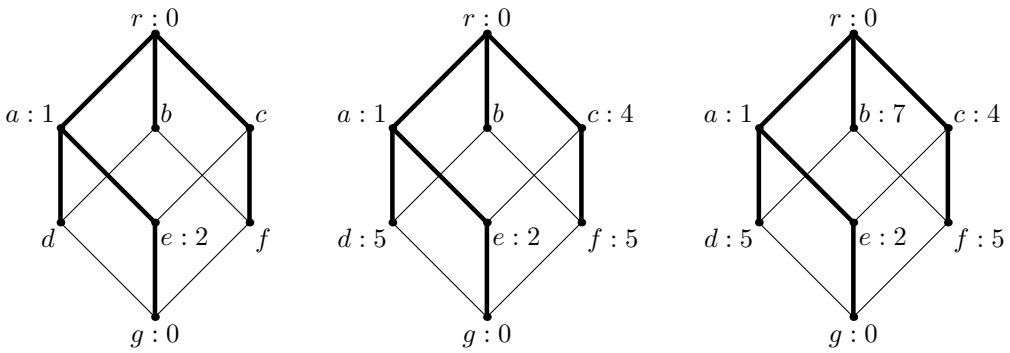
4. u sends PUT-COL(v,k_v) to v (and all stepchildren of u), where k_v is the smallest color in the palette of v not in $K \cup L$.

5. v broadcasts RPT-COL(k_v,v).

6. Each child of v sends RPT-PAR(k_v,v) to all its stepparents.

In this sketch, for the sake of simplicity we have ignored the DEP facility and the problem of dependency cycles in general.

Observe that the set of messages consumed by any of the three procedures is disjoint from the other two: AcquireColor listens for RPT-PAR from



(a) a, e, g request and receive colors

(b) c, d, f request and receive colors

(c) b requests and receives a color

Figure 8: The 3-band algorithm on the 3-cube Q_3; the spanning tree is indicated by thick edges.

(step)children, DEP-REQ from children, and PUT-COL from the parent; AssignColors listens for REQ-COL from children, RPT-COL from (step)children, and DEP-PUT from children; and ReportColors listens for DEP-REQ from parents, RPT-PAR/DEP-PUT from (step)parents, and PUT-COL from (some) stepparents. Therefore each procedure may be assumed to operate on its own received message queue independently from the others.

In general, the d-band algorithm colors the leaves of T first and proceeds towards the root, although significant parallelism is possible. Refer to Figure 8 for an example.

3.4.1 Proof of correctness

We say that the d-band algorithm terminates on graph G with r-rooted spanning tree T iff AcquireColor returns on every vertex of G, a condition equivalent to the coloring being a total function. Unlike our previous algorithms, it is not at all obvious that the d-band algorithm terminates, so we must prove it explicitly.

AcquireColor on vertex v does its main work in the loop beginning at line 6. As this loop is bypassed when v has no stepchildren, assume that it does. We say that v has a *request dependence* on u when u is the parent of a stepchild of

v; and just as v depends on u, u may depend on t, and so on. If we can follow the dependency chain to some terminal a that has no stepchildren, there is no problem, since we can inductively work back to v. However, the dependency chain may in fact be a cycle, in the sense that v has request dependence on u, u has request dependence on t, and so on up to a, but then a has request dependence on v. This we call a *dependency cycle of type I*.

Given C, a dependency cycle of type I, let $v = \min_{\prec} C$. Assume, for the time being, that C is the only dependency cycle in the graph. v issues DEP-REQ(v) to its children, and via ReportColors one of the children sends RPT-PAR(∞,v) to x, which depends on v. But since $v \prec x$, x issues DEP-REQ(v) to its children, one of which then sends RPT-PAR(∞,v) to y, which depends on x, and so on.

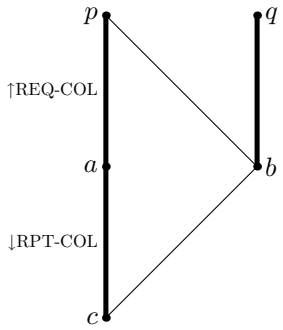
Let u be the vertex in C on which v depends, creating the cycle. u issues DEP-REQ(v) to its children, and one of them transmits RPT-PAR(∞,v) to v. At this point v can detect the dependency cycle, and v breaks the cycle by ignoring its dependence on u (see line 12 of AcquireColor). As per our assumptions, v is now free of request dependencies, or at worst sits in linear depedence chains that are naturally resolved; that is, v (eventually) acts as if it has no stepchildren, and proceeds to issue REQ-COL to its parent p.

Let us assume that p eventually assigns a color to v via PUT-COL. v then broadcasts RPT-COL, resolving the now-linear dependency chain. (The resolution is a little unusual at u, where we have registered a "reverse dependence" on v—but this will be cleared by the RPT-COL broadcast from v, which causes a "reverse report" RPT-PAR to be sent to u from one of its children.) Hence every vertex in C gradually becomes free to issue REQ-COL, and if we assume that every one of their parents replies with PUT-COL, then AcquireColor terminates on every vertex in C.

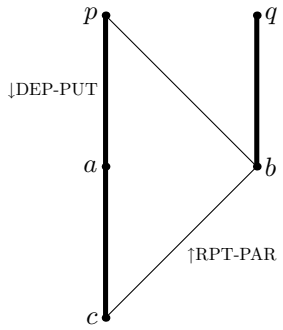
We now shift our attention to AssignColors. A vertex v with parent p_v is said to have a *put dependence* on any stepchild of p_v. (It is convenient for the dependence to be registered at p_v.) Just like request dependencies, put dependencies can be chained and may form a cycle; this we call a *dependency cycle of type II*. It is not hard to see that a type II cycle is broken by essentially the same method used for the type I cycle, with DEP-PUT and RPT-COL standing in for DEP-REQ and RPT-PAR, respectively. (Once again, there is a special "reverse dependence" facility. Let p_v have stepchild u with parent p_u. If a type II cycle is broken at p_v, then p_u will register the reverse dependence of u on the children of p_v. Resolution comes when p_u finishes coloring its children, with "reverse report" RPT-COLs being sent to p_u via u.) Hence the d-band algorithm terminates in the presence of a dependency cycle of type II, provided that REQ-COL is issued.

Finally, a dependency cycle of *mixed type* is possible. AcquireColor handles the transition from request to put dependency by repackaging RPT-PAR as RPT-COL (line 20), while ReportColors handles the reverse transition by first repackaging DEP-PUT as RPT-COL (line 17) and then relaying the latter as RPT-PAR (line 11). But observe that a cycle of mixed type can be broken by the methods previously described; it is treated exactly as if it were a cycle of type I or type II if it is broken by AcquireColor or AssignColors, respectively. The same mechanisms then assure that the resolution proceeds across the cycle. See Figure 9 for a simple example. We conclude that the d-band algorithm terminates if there is no more than one dependency cycle in the graph (of whatever type).

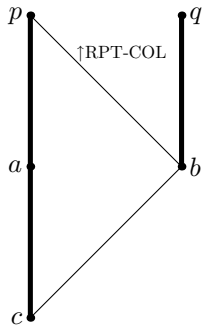
Given a dependency cycle C (of any type), observe that there exists some l such that $\text{dist}_T(v, r) = l$ for all $v \in C$; call l the level of C. Clearly cycles with different levels cannot affect each other; additionally, disjoint cycles do not interact. Thus the d-band algorithm terminates given any number of disjoint dependency



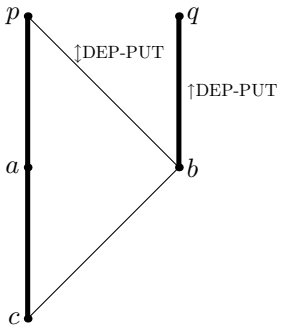
(a) a sends RPT-COL(∞,a) to a and REQ-COL($\emptyset$) to c



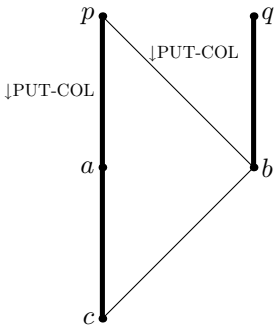
(b) p sends DEP-PUT(b) to a; c sends RPT-PAR(∞,a) to b



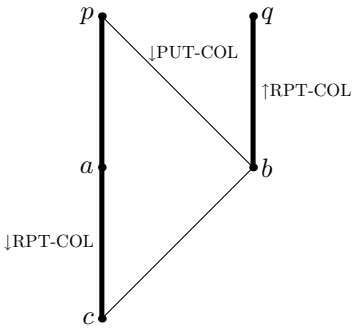
(c) b sends RPT-COL(∞,a) to p



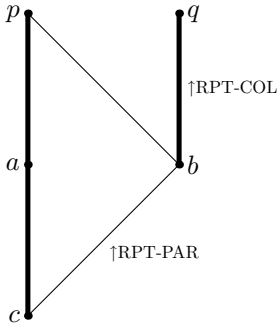
(d) p detects and breaks the dependency cycle: p ack-sends DEP-PUT(p) to b; b forwards DEP-PUT(p) to q



(e) p sends PUT-COL(a,α) to a, b



(f) p sends PUT-COL(p,∞) to b; a sends RPT-COL(α,a) to c; b sends RPT-COL(α,p) to q



(g) b sends RPT-COL(∞,p) to q; c sends RPT-PAR(α,a) to b; all dependencies are resolved

Figure 9: Resolving of a dependency cycle of mixed type; the spanning tree is indicated by thick edges. Assume a is the $\prec$-smallest vertex.

cycles per level of T.

Unfortunately a graph may contain many overlapping dependency cycles—yet we claim the algorithm terminates regardless. Let $\mathcal{C}$ be a family of intersecting dependency cycles. As there is a strict total order $\prec$ on vertices, there exists some

$$v = \min_{\prec} \bigcup_{C \in \mathcal{C}} C.$$

Observe that AcquireColor must terminate on v, since all dependency cycles in $\mathcal{C}$ containing v will be broken at v, if not elsewhere. After breaking all such cycles and resolving all newly-linear chains, let $\mathcal{C}'$ be the remaining cycles. Obviously $|\mathcal{C}'| < |\mathcal{C}|$, and we can apply the same argument to $\mathcal{C}'$ inductively. We conclude that the d-band algorithm terminates.

At this point we have shown that the coloring function produced by the d-band algorithm is total. We now prove that it is a T-pseudo-schedule.

Theorem 3.6. *Let G be a graph with spanning tree T rooted at r. The d-band algorithm produces a T-pseudo-schedule provided that*

$$d \geq \min \left(\operatorname{height}(T) + 1, \max_{uv \in E(G)} |\operatorname{dist}_T(u, r) - \operatorname{dist}_T(v, r)| + 2 \right).$$

Proof. Let s be the coloring produced by the algorithm. Consider $uv \in E(T)$ such that u is the parent of v; first we establish that (u, v) is nonconflicting. Observe that we only need consider the stepparents of v, for any other vertices in $N_G[v]$ are colored from a different palette than u, given our choice of d. In general a stepparent t of v waits until u is colored before requesting a color from its parent p_t, precisely in order to exclude $s(u)$. This will not obtain if a dependency cycle is broken by AcquireColor at t; but in this case, v issues DEP-REQ to u, causing u to register a "reverse dependence" on t. The RPT-COL of t announcing its color is forwarded by v to u as a "reverse report" RPT-PAR, and u will add $s(t)$ to its

set of excluded colors when it requests a color from its parent. So even in this case, $s(u) \neq s(t)$.

Now consider (v, u). Observe that we only need consider the children and stepchildren of u, for any other vertices in $N_G[u]$ are colored from a different palette than v per our choice of d. Now any child $x \neq v$ of u must have $s(x) \neq s(v)$ since u assigns the colors of its children. In general u also waits for its stepchildren to be colored before assigning colors to its children, but this would not have been the case if a dependency cycle was broken by AssignColors at u. However, when breaking the dependency cycle, u issues DEP-PUT to its stepchildren, and any such stepchild y forwards DEP-PUT to its parent p_y, which registers a "reverse dependence" of y on the children of u. Every PUT-COL issued by u is subsequently copied to y, which sends a "reverse report" RPT-COL to p_y. In this way p_y comes to know the colors of the children of u, and will subsequently assign a color to y distinct from those. This implies $s(v) \neq s(y)$. □

It is clear from the preceding theorem that $d \geq 3$. Requiring $d = 3$ is useful and not, in practice, very restrictive, since the 3-band algorithm works on trees that minimize distance to the root.

Corollary 3.7. *If T is an r-rooted spanning tree of G such that $\mathrm{dist}_T(v, r) = \mathrm{dist}_G(v, r)$ for any $v \in V(G)$, then the 3-band algorithm produces a T-pseudo-schedule.*

Finally, suppose G is such that for any $uv \in E(G) - E(T)$, $|\,\mathrm{dist}_T(u, r) - \mathrm{dist}_T(v, r)| \neq 1$. In this case dependency cycles cannot occur at all, and the algorithm becomes almost trivial!

3.4.2 Analysis

The analysis of the d-band algorithm is blessedly simpler than the proof of correctness. For this section, we say that a d-band coloring s uses $(\max \text{Image}(s)) + 1$ colors; this forces us to account for "gaps" in the set of colors actually appearing in s.

Theorem 3.8. *Let h denote the number of colors used by the d-band algorithm on graph G with spanning tree T rooted at r, where d meets the conditions of Theorem 3.6. If $d \leq \text{height}(T) + 1$ and $\Delta_G \geq 2$, then*

$$d(\Delta_T - 1) + 1 \leq h \leq 2d(\Delta_G - 1)$$

except possibly when $\Delta_T = 2$, in which case the lower bound falls to d. These bounds are tight.

Proof. The lower bound for $\Delta_T = 2$ is established by any path graph. For $\Delta_T \geq 3$, let v be the vertex on which T achieves its maximum degree. Then any child of v uses a palette containing at least the colors $a, a + d, \ldots, a + d(\Delta_T - 1)$. With $a = 0$, we obtain the lower bound. To see tightness, consider a path graph of order d with r at one end; to the other end, attach m vertices, where $m \geq 2$. The pseudo-schedule on this tree uses $dm + 1$ colors and has degree $m + 1$.

For the upper bound, consider a vertex v with $\text{dist}_T(v, r) = d - 1$. Its parent p_v has at most $\Delta_G - 2$ neighbors distinct from v but at the same T-level as v. Additionally, v has at most $\Delta_G - 1$ stepchildren, each of which could have a distinct parent. This is a total of $2\Delta_G - 2$ vertices (including v), so

$$\text{Palette}(v) \subseteq \{d - 1, 2d - 1, \ldots, d - 1 + d(2\Delta_G - 3)\}$$

which yields the upper bound.

To establish tightness, first construct T as follows: fix some $m \geq 2$, introduce vertex r, and attach to it two paths of length $d-2$ each. Let the endpoints be called

p_x and p_y, and attach to each the vertices $x_1, x_2, \ldots, x_{m-1}$ and $y_1, y_2, \ldots, y_{m-1}$ respectively. Finally, to each y_i attach a vertex z_i $(1 \leq i \leq m-1)$.

To build G, take T and introduce the edges $x_i z_j$ for every $1 \leq i, j \leq m-1$. T is a spanning tree of G and $\Delta_G = \Delta_T = m$. Furthermore, it is easy to confirm that every vertex in $\{x_1, y_1, x_2, y_2, \ldots, x_{m-1}, y_{m-1}\}$ must be colored differently under any T-pseudo-schedule. Since these vertices are all at T-level $d-1$, we hit the upper bound. □

In general we obtain an approximation ratio of $\frac{2}{3}d\Delta$.

3.4.3 Implementation issues

The worst-case number of colors used by the d-band algorithm is, importantly, linear in the degree of the graph, but typically one should expect performance worse than either the greedy or 2Δ algorithms due to the potentially large gaps in the image of the coloring function. We do recover complete freedom in the choice of spanning tree, but with potentially explosive consequences for d; in practice one must use spanning trees that (nearly) minimize distance to root. Conceptually the algorithm is qualitatively more complex than the alternatives due to the problems of resolving dependencies.

That said, the d-band algorithm is almost surely the best protocol candidate of the three algorithms we have encountered. Reliance on information at distance two is quite modest thanks to the division of the graph into bands that operate virtually independently. Most importantly, the algorithm is distributed and decentralized, requiring no mutual exclusion mechanisms: determining when to "wait" flows naturally out of the algorithm's state semantics. Very aggressive parallelism is also possible, although this is spoiled by the presence of dependency chains and cycles. As a rule of thumb, spanning trees should be constructed (more or less) greedily to avoid such cycles—but of course this tends to boost Δ_T, which in turn

tends to increase color consumption.

Finally, the d-band algorithm is an example of *self-organization* as defined by Dressler [6]:

> Self-organization is a process in which structure and functionality (pattern) at the global level of a system emerge solely from numerous interactions among the lower-level components of a system without any external or centralized control. The systems components interact in a local context either by means of direct communication or environmental observations without reference to the global pattern.

Indeed, the d-band algorithm is arguably a very strong example of a self-organizing system, since its global structure is provably, not just heuristically or empirically, guaranteed.[2] The ideas realized in the design of the algorithm may be useful in the general study of self-organization.

3.5 Dismantleable (cop-win) graphs

Recall that a vertex u *dominates* a vertex v in G iff $N_G[v] \subseteq N_G[u]$. A graph is *dismantleable* iff it is trivial or there exists a dominated vertex v such that $G - v$ is dismantleable. Dismantleable graphs are also called *cop-win* because they are precisely the graphs on which the following vertex pursuit game, played by a "cop" and a "robber," can always be won by the cop: the cop chooses a starting vertex, followed by the robber; the players move alternately, starting with the cop, with each move consisting of staying put or going to an adjacent vertex. The game is played with complete information, and the cop wins iff he moves into the vertex occupied by the robber. The equivalence of the two classes was discovered by Nowakowski and Winkler [7].

A third characterization of dismantleable graphs is due to Prisner [8]. For any graph G, let the *pared graph* $P(G)$ be defined as follows: first, identify all vertices

[2]One might object that the parameter d relies on global information, which is true in principle. However, with $d = 3$ the user need only provide a spanning tree minimizing distance to the root, which can itself be constructed in a self-organized way. The order $\prec$ on vertices is also global, but it is almost always provided "for free" in practice; eg, by serial numbers or MAC addresses.

with the same closed neighborhood with a single representative; then remove all dominated vertices. We can pare the graph as many times as we like, so let the kth pared graph $P^k(G)$ be defined as $P(P^{k-1}(G))$ where $P^0(G) = G$. A graph is dismantleable iff there exists some nonnegative integer k such that the kth pared graph is trivial; the smallest such k is called the *pare-index* of G.

Via the pared graphs, any dismantleable graph has a natural spanning tree: for any vertex v in $P^k(G)$ but not $P^{k+1}(G)$, let the parent of v be a vertex that dominates it in $P^k(G)$. Observe that the height of the resulting spanning tree is equal to the pare-index of G.

Let a dismantleable graph G have r-rooted spanning tree T as just described, and let $\mathrm{dom}(v)$ denote the parent of v in T. Algorithm 3.5.1 produces a T-pseudo-schedule on G. We require some notation: a vertex u is *prior* to v iff u is adjacent to $\mathrm{dom}(v)$. We write $u \preceq v$ iff u is prior to v, or u is prior to some x such that $x \preceq v$; observe that $\preceq$ is a preorder (a reflexive, transitive binary relation). Finally, $u \sim v$ iff $u \preceq v$ and $v \preceq u$. If we let $V_k = \{v \in V(G) : \mathrm{dist}_T(r, v) = k\}$, observe that $\sim$ partitions V_k and that the blocks of $V_k/\sim$ have a natural total strict order (possibly by linear extension).

3.5.1 Proof of correctness

We show that the coloring s produced by Algorithm 3.5.1 on a dismantleable graph G, given a dom spanning tree T rooted at r, is a T-pseudo-schedule. Note that V_k is colored before V_{k+1} for all $k \geq 0$. Let $uv \in E(T)$ such that $u = \mathrm{dom}(v)$. First we show that (u, v) is nonconflicting. Clearly $s(u) \neq s(v)$. Now for any $x \in N_G[v]$ distinct from u, v, we have $\mathrm{dist}_T(r, x)$ within one of $\mathrm{dist}_T(r, v)$. Suppose $\mathrm{dist}_T(r, x) < \mathrm{dist}_T(r, v)$; since u dominates v, x is adjacent to u. But then $\mathrm{dom}(u)$ is adjacent to x and $\mathrm{dom}(x)$ is adjacent to u, so $s(u) \neq s(x)$, regardless of which is colored first. If, alternately, the distances are equal, then x is adjacent to u, thus

Algorithm 3.5.1: Pseudo-scheduling the dismantleable graph

Input: G, a dismantleable graph with dom spanning tree T rooted at r.
Output: A T-pseudo-schedule on G.

```
1  s ← {r ↦ 1}
2  for k ← 1 ... pare-index of G do
3      foreach B ∈ V_k/∼ (in natural order) do
4          foreach v ∈ B do
5              if s(v) undefined then
6                  P_v ← {x ∈ B; x prior to v}            // Note v ∈ P_v
7                  foreach p ∈ P_v do
8                      K ← s(N_G[dom(p)]) ∪ s(N_T[dom(v)] ∩ B)
9                      k ← min(ℤ⁺ − K)
10                     add p ↦ k to s
11 return s
```

dom(x) is adjacent to u, and $s(u) \neq s(x)$. In the final case, dom(x) is adjacent to v, hence u is adjacent to dom(x), and $s(u) \neq s(x)$.

Now consider (v, u). For any $w \in N_G[u]$ distinct from u, v, we have $\text{dist}_T(r, w)$ within one of $\text{dist}_T(r, u)$. If the former is no more than the latter, $s(v) \neq s(w)$ is obvious, since w was colored first. In the remaining case, note that w is prior to v. If w was colored first, $s(v) \neq s(w)$; if, however, v was colored first, it must be that v, w are in the same block $B \in V_{\text{dist}_T(r,v)}/\sim$. But then w could not have been given the same color as any T-child of u in B. We conclude that $s(v) \neq s(w)$.

3.5.2 Analysis

Theorem 3.9. *Let h denote the number of colors used by Algorithm 3.5.1 on a nontrivial dismantleable graph G with* dom *spanning tree T rooted at r. Then*

$$\Delta_T + 1 \leq h \leq \Delta_G + \Delta_T.$$

Furthermore, these bounds are tight.

Proof. The lower bound and its tightness are obvious. The upper bound comes immediately from counting the number of examined vertices (see line 8 of the

algorithm).

To see that the upper bound is tight, begin with K_4: let one vertex be r, and call the other vertices p_x, p_y, p_z. Fix $m \geq 2$. Attach to p_x the vertices $x_1, x_2, \ldots, x_m$, to p_y the vertices $y_1, y_2, \ldots, y_m$, and to p_z the vertices $z_1, z_2, \ldots, z_m$. Finally, introduce the edges $x_i p_y, y_i p_z, z_i p_x$ for all $1 \leq i \leq m$. This graph has a dom spanning tree as such: $\mathrm{dom}(p_x) = \mathrm{dom}(p_y) = \mathrm{dom}(p_z) = r$, $\mathrm{dom}(x_i) = p_x$, $\mathrm{dom}(y_i) = p_y$, and $\mathrm{dom}(z_i) = p_z$. There are $3m + 4$ vertices, and it is not hard to see that each one receives a unique color. The degree of the graph is $2m + 3$, and the degree of the spanning tree is $m + 1$, so the upper bound is achieved. □

The improvement in colors over the 2Δ algorithm is only marginal, but here we have the advantage of a spanning tree with rich semantic content, as opposed to one selected for the algorithm's peculiar advantage. We could, for example, set up a detection network modeled on a dismantleable graph to direct an automaton intercepting an intruder: the "cop-win" structure guarantees interception eventually—in number of turns not exceeding the pare-index, in fact—while network communications are organized by means of the st-pseudo-schedule.

List of References

[1] T. Ren, "Graph coloring algorithms for TDMA scheduling in wireless sensor networks," Ph.D. dissertation, University of Rhode Island, 2007.

[2] T. Calamoneri, "The $L(h, k)$-labelling problem: A survey and annotated bibliography," *The Computer Journal*, vol. 49, no. 5, pp. 585–606, 2006.

[3] S. T. McCormick, "Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem," *Mathematical Programming*, vol. 26, pp. 153–171, 1983.

[4] M. Fürer and B. Raghavachari, "Approximating the minimum-degree Steiner tree within one of optimal," *Journal of Algorithms*, vol. 17, no. 3, pp. 409–423, 1994.

[5] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks," in *Seventh ACM International*

Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006), May 2006, pp. 190–201.

[6] F. Dressler, "A study of self-organization mechanisms in ad hoc and sensor networks," *Computer Communications*, vol. 31, no. 13, pp. 3018–3029, 2008.

[7] R. Nowakowski and P. Winkler, "Vertex-to-vertex pursuit in a graph," *Discrete Mathematics*, vol. 43, pp. 235–239, 1983.

[8] E. Prisner, "Convergence of iterated clique graphs," *Discrete Mathematics*, vol. 103, pp. 199–207, 1992.

Algorithm 3.5.2: AcquireColor

Input: v, "this" vertex.
Output: A color.

```
1  L ← ∅
2  f_W ← {x ↦ v; x is a stepchild of v}
3  W ← Image(f_W)
4  w_≺ ← v
5  send RPT-COL(∞,v) to children of v
6  while W ≠ ∅ do
7      listen for message M
8      if M = RPT-PAR(k,w) from (step)child x of v then
9          if k ≠ ∞ then
10             remove x ↦ f_W(x) from f_W
11             add k to L
12         else if w = v then remove x ↦ f_W(x) from f_W
13         else add/replace x ↦ w in f_W
14         W ← Image(f_W)
15     else if M = DEP-REQ(w) from a child of v then
           // Reverse dependence
16         add/replace w ↦ w in f_W
17     if w_≺ ≠ min_≺(W ∪ {v}) then
18         w_≺ ← min_≺(W ∪ {v})
19         send DEP-REQ(w_≺) to children of v
20         send RPT-COL(∞,w_≺) to stepparents of v
21 send RPT-COL(∞,v) to children of v
22 send REQ-COL(L) to the parent of v
23 listen for PUT-COL(v,k_v) from the parent of v
24 send RPT-COL(k_v,v) to children, stepchildren, and stepparents of v
25 return k_v
```

Algorithm 3.5.3: AssignColors

Input: v, "this" vertex.

```
1  K, f_L, f_W, W ← ∅
2  X ← stepchildren of v, Z ← children of v
3  f_R ← {z ↦ ∅; z ∈ Z}
4  w_≺ ← v
5  while Z ≠ ∅ do
6      if X = ∅ then
7          foreach z ↦ L ∈ f_L such that f_R(z) = ∅ do
8              k_z ← min(Palette(z) − K − L)
9              send PUT-COL(z,k_z) to z and stepchildren of v
10             remove z from Z
11             add k_z to K
12     else if W ≠ ∅ and w_≺ ≠ min_≺ W then
13         w_≺ ← min_≺ W
14         send DEP-PUT(w_≺) to children of v
15     listen for message M
16     if M = REQ-COL(L) from child z of v then
17         if f_L(z) defined then L ← L ∪ f_L(z)
18         add/replace z ↦ L in f_L
19     else if M = RPT-COL(k,w) from (step)child x then
20         if x ∈ X then
21             if k ≠ ∞ or w is a child of v then
22                 remove x from X
23                 remove x ↦ f_W(x) from f_W
24                 add k to K
25                 if w is a child of v then ack-send DEP-PUT(v) to x
26             else add/replace x ↦ w to f_W
27             W ← Image(f_W)
28         else if x is a child of v then
29             if k ≠ ∞ then
30                 L ← (f_L(x) defined ? f_L(x) : ∅)
31                 add k to L
32                 add/replace x ↦ L in f_L
33             else remove w from f_R(z)
34     else if M = DEP-PUT(w) from child z of v then
           // Reverse dependence
35         add w to f_R(z)
36 send PUT-COL(∞,v) to stepchildren of v
```

Algorithm 3.5.4: ReportColors

```
   Input: v, "this" vertex.
 1 p ← parent of v
 2 W_I, W_II ← ∅
 3 listen for message M from (step)parents
 4 if M = DEP-REQ(w) from p then
 5 |   if w is a stepparent of v then
   |   |   // Type I cycle breaking:  reverse dependence
 6 |   |   add w to W_I
 7 |   |   send DEP-REQ(w) back to p
 8 |   send RPT-PAR(∞,w) to stepparents of v
 9 else if M = RPT-COL(k,w) from x then
10 |   if x = p then
11 |   |   send RPT-PAR(k,w) to stepparents of v
12 |   else if x ∈ W_I and k ≠ ∞ then
   |   |   // Type I cycle breaking:  reverse report
13 |   |   remove x from W_I
14 |   |   send RPT-PAR(k,x) to p
15 else if M = DEP-PUT(w) from x then
16 |   if x = p then
17 |   |   send RPT-COL(∞,w) to children and stepparents of v
18 |   else
   |   |   // Type II cycle breaking:  reverse dependence
19 |   |   add x to W_II
20 |   |   send DEP-PUT(x) to p and x
21 else if M = PUT-COL(u,k) from x ∈ W_II then
   |   // Type II cycle breaking:  reverse report
22 |   if k = ∞ then remove x from W_II
23 |   send RPT-COL(k,x) to p
```

CHAPTER 4

Hardness

In this chapter we consider the *hardness of optimization* (more precisely, the hardness of the related decision problem) and the *hardness of approximation* of pseudo-scheduling algorithms.

Let PSCHED be the set of ordered pairs (G, d) such that

$$(G, d) \in \text{PSCHED} \iff \widetilde{\chi}_{1,1}(G) \leq d.$$

The set can be described as a language of strings by canonical encoding methods. We can also define the analogous set/language STPSCHED where

$$(G, d) \in \text{STPSCHED} \iff \widehat{\chi}_{1,1}(G) \leq d.$$

It is easy to prove that both sets are in NP.

Lemma 4.1. PSCHED *and* STPSCHED *are in the complexity class NP.*

Proof. Given a pair (G, d), let $s : V(G) \to K$ be an alleged pseudo-schedule on G acting as witness. It is easy to confirm in polynomial time that s is a coloring function and that $|K| \leq d$. Now given any vertex v in G, one can determine by a BFS-like algorithm the set of vertices reachable from v by a nonconflicting path and check that this includes all other vertices. We do this for every vertex, with the entire process being completed in polynomial time. If all checks are passed, then s certifies that $\widetilde{\chi}_{1,1}(G) \leq d$.

For STPSCHED, we require as witness, in addition to s, a spanning tree T. We confirm that T is a spanning tree and that for each $e \in E(T)$, e is bidirectional under s—all of which can be done in polynomial time. If the checks are passed, then s is a T-pseudo-schedule, thus an st-pseudo-schedule. □

The main results of this chapter are presented below.

Theorem 4.2. PSCHED *and* STPSCHED *are NP-Complete.*

An algorithm $\mathcal{A}$ is a *fully polynomial-time approximation scheme (FPTAS)* for PSCHED (STPSCHED) iff for any graph G and $\epsilon > 0$, $\mathcal{A}$ can produce a (spanning tree) pseudo-schedule on G using no more than $(1+\epsilon)\widetilde{\chi}_{1,1}(G)$ $((1+\epsilon)\widehat{\chi}_{1,1}(G))$ colors in time polynomial in the size of G and $1/\epsilon$.

Theorem 4.3. *Neither* PSCHED *nor* STPSCHED *admit an FPTAS, unless* P = NP.

4.1 Hardness of optimization

Our method of proving that PSCHED is NP-Hard will be analogous to Karp's proof of the hardness of $L(1)$-labeling [1] and McCormick's proof of the hardness of $L(1,1)$-labeling [2]; that is, reduction from 3SAT, the language of satisfiable Boolean formulas in conjunctive normal form with exactly three literals per clause.

It is convenient to describe 3SAT in set-theoretic terms. Given a set of n *variables* $\{x_1, \ldots, x_n\}$, we can give a set of $2n$ *literals* $\{x_1, \overline{x_1}, \ldots x_n, \overline{x_n}\}$. A *clause* is a three-element subset of the literals. A *formula* is a set of m clauses $\{C_1, \ldots, C_m\}$. Then a formula $\mathcal{F}$ on variables X is in 3SAT iff there exists some *truth assignment* $\phi : X \to \{\top, \bot\}$ such that

$$\forall C \in \mathcal{F}, \exists x \in X : (x \in C \wedge \phi(x) = \top) \vee (\overline{x} \in C \wedge \phi(x) = \bot).$$

We make the following assumptions about $\mathcal{F}$:

1. No clause of $\mathcal{F}$ contains both x and $\overline{x}$ for any variable x.

2. For every variable x, both x and $\overline{x}$ appear in different clauses of $\mathcal{F}$.

These assumptions hold without loss of generality, as any well-formed formula can be converted into the desired form in polynomial time.

Our aim is to exhibit, for any formula $\mathcal{F}$ with n variables and m clauses, a graph $G(\mathcal{F})$ that admits an (st-)pseudo-schedule in $m+n+5$ colors iff $\mathcal{F} \in 3\text{SAT}$.

The first element of $G(\mathcal{F})$ is the *color gadget* $\mathfrak{K}_{m,n}$ where

$$\begin{aligned} V(\mathfrak{K}_{m,n}) &= \{k_i; 0 \leq i \leq m+n+4\} \\ &\quad \cup \{k_{i,j}; 0 \leq i < j \leq m+n+4\}, \\ E(\mathfrak{K}_{m,n}) &= \{\{k_i, k_{i,j}\}, \{k_{i,j}, k_j\}; 0 \leq i < j \leq m+n+4\}. \end{aligned}$$

Note that K_{m+n+5} is a graph minor of $\mathfrak{K}_{m,n}$ by contraction of the edges $\{k_i, k_{i,j}\}$.

Also comprising $G(\mathcal{F})$ are the *variable gadgets* $\mathfrak{X}_i$ where

$$\begin{aligned} V(\mathfrak{X}_i) &= \{x_i, \overline{x_i}, z_i\} \\ &\quad \cup \{x_{i,j}; 1 \leq j \leq m+n+4, j \neq i+2\} \\ &\quad \cup \{\overline{x_{i,j}}; 1 \leq j \leq m+n+4, j \neq i+2\}, \\ E(\mathfrak{X}_i) &= \{\{x_i, z_i\}, \{\overline{x_i}, z_i\}\} \\ &\quad \cup \{\{x_i, x_{i,j}\}; x_{i,j} \in V(\mathfrak{X}_i)\} \\ &\quad \cup \{\{\overline{x_i}, \overline{x_{i,j}}\}; \overline{x_{i,j}} \in V(\mathfrak{X}_i)\} \end{aligned}$$

for $1 \leq i \leq n$.

The final component of $G(\mathcal{F})$ are the vertices $\{C_1, \ldots, C_m\}$, which represent the clauses.

We can now completely define the graph $G(\mathcal{F})$. The vertex set is given by

$$\begin{aligned} V(G(\mathcal{F})) &= V(\mathfrak{K}_{m,n}) \\ &\quad \cup V(\mathfrak{X}_1) \cup \cdots \cup V(\mathfrak{X}_n) \\ &\quad \cup \{C_1, \ldots, C_m\}. \end{aligned}$$

The edge set is given by

$$\begin{aligned} E(G(\mathcal{F})) = E(\mathfrak{K}_{m,n}) \\ &\cup E(\mathfrak{X}_1) \cup \cdots \cup E(\mathfrak{X}_n) \\ &\cup \{\{k_j, x_{i,j}\}; 1 \leq i \leq n, 1 \leq j \leq m+n+4, j \neq i+2\} \\ &\cup \{\{k_j, \overline{x_{i,j}}\}; 1 \leq i \leq n, 1 \leq j \leq m+n+4, j \neq i+2\} \\ &\cup \{\{x_i, C_j\}; 1 \leq i \leq n, 1 \leq j \leq m, x_i \in C_j\} \\ &\cup \{\{\overline{x_i}, C_j\}; 1 \leq i \leq n, 1 \leq j \leq m, \overline{x_i} \in C_j\}. \end{aligned}$$

Figure 10 gives an example of the global structure of $G(\mathcal{F})$ about a single variable gadget.

The first step in exhibiting a reduction from 3SAT is showing that some satisfying assignment of a formula can be translated into a (st-)pseudo schedule on the corresponding graph.

Lemma 4.4. *If $\mathcal{F} \in$ 3SAT, then there exists an (spanning tree) pseudo-schedule of $G(\mathcal{F})$ in $m+n+5$ colors.*

Proof. Let K consist of one "false color" $\bot$; n "true colors" $\top_1, \ldots, \top_n$; m "clause colors" $\alpha_1, \ldots, \alpha_m$; and four "separator colors" $\omega_1, \ldots, \omega_4$. We shall show that there exists an st-pseudo-schedule $s : V(G(\mathcal{F})) \to K$ provided there exists a truth assignment ϕ satisfying $\mathcal{F}$.

We start by defining s over the color gadget, beginning with the following

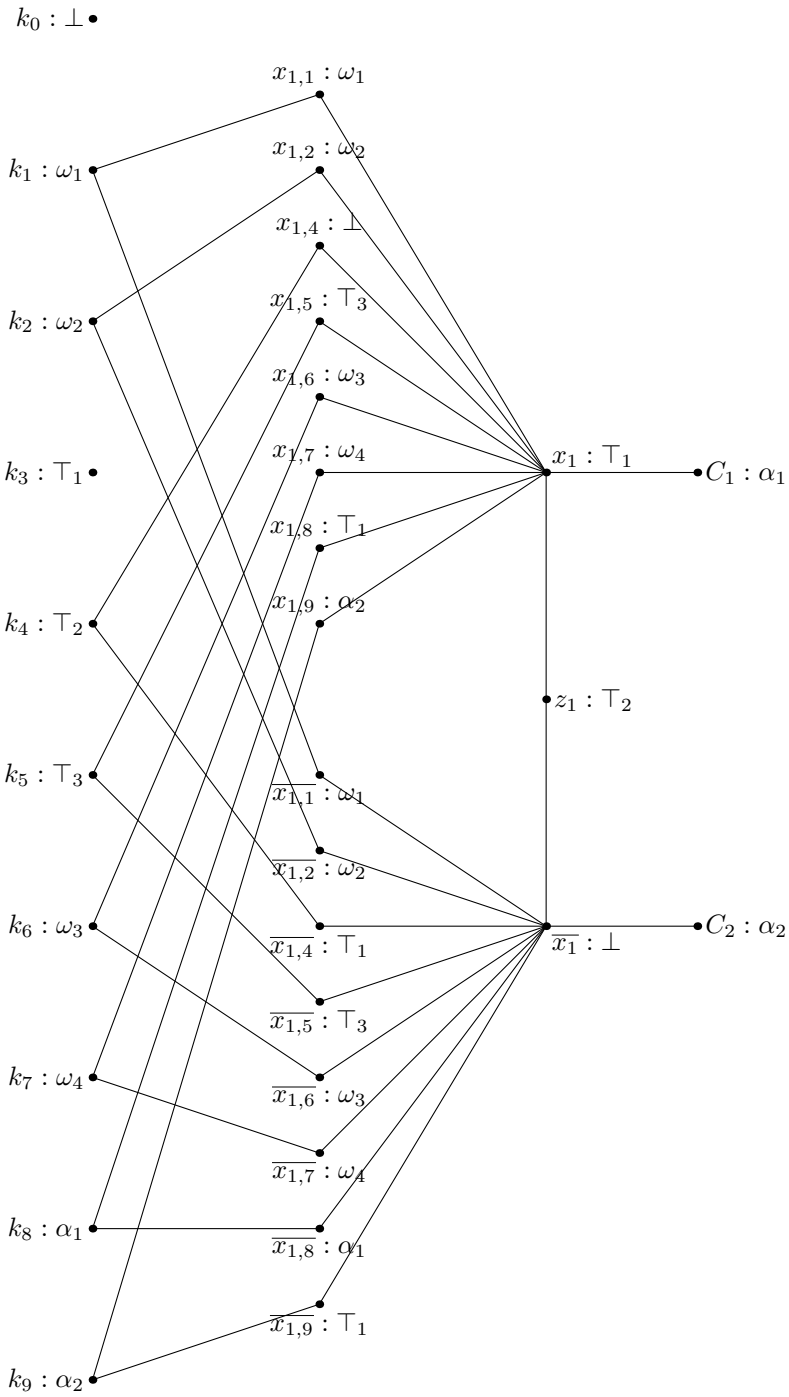


Figure 10: Structure of $G(\mathcal{F})$ for the formula $\mathcal{F} = \{\{x_1, \overline{x_2}, x_3\}, \{\overline{x_1}, x_2, \overline{x_3}\}\}$ about the variable gadget $\mathfrak{X}_1$. Colors are assigned as per the proof of Lemma 4.4, assuming a truth assignment where $x_1 \mapsto \top$.

assignments:

$$k_0 \mapsto \bot$$

$$k_1 \mapsto \omega_1$$

$$k_2 \mapsto \omega_2$$

$$k_{i+2} \mapsto \top_i \quad (1 \leq i \leq n)$$

$$k_{n+3} \mapsto \omega_3$$

$$k_{n+4} \mapsto \omega_4$$

$$k_{j+n+4} \mapsto \alpha_j \quad (1 \leq j \leq m).$$

For the next set of assignments, we will find it convenient to let $k_{b,a}$ be an alias for $k_{a,b}$ when $a < b$; we also consider all subscripts to be modulo $m + n + 5$. Furthermore, we allow ourselves to refer to the color $s(v)$ whenever s is already defined on the vertex v. Let us continue, then, with the vertices on the major cycle:

$$k_{i,i+1} \mapsto s(k_{i+2}) \quad (0 \leq i \leq m + n + 4)$$

and then the cycle in steps of two:

$$k_{i,i+2} \mapsto s(k_i) \quad (0 \leq i \leq m + n + 4).$$

To complete the color gadget, we introduce the mappings

$$k_{i,j} \mapsto \begin{cases} s(k_j) = \alpha_{j-n-4}, & \text{if } n + 5 \leq j \leq m + n + 4 \\ s(k_i), & \text{otherwise} \end{cases}$$

for all $0 \leq i < j \leq m + n + 4$ where $k_{i,j}$ has not yet been colored.

We now turn to the variable gadgets. For all $1 \leq i \leq n$, color $\mathfrak{X}_i$ as follows:

$$x_i \mapsto \begin{cases} \top_i, & \text{if } \phi(x_i) = \top \\ \bot, & \text{otherwise} \end{cases}$$

$$x_{i,1} \mapsto \omega_1$$

$$
\begin{aligned}
x_{i,2} &\mapsto \omega_2 \\
x_{i,j} &\mapsto s(k_j) \quad (3 \le j < i+2) \\
x_{i,i+3} &\mapsto \begin{cases} \top_i, & \text{if } \phi(x_i) = \top \\ \bot, & \text{otherwise} \end{cases} \\
x_{i,j} &\mapsto s(k_j) \quad (i+4 \le j \le n+2) \\
x_{i,n+3} &\mapsto \omega_3 \\
x_{i,n+4} &\mapsto \omega_4 \\
x_{i,j+n+4} &\mapsto \begin{cases} \top_i, & \text{if } x_i \in C_j \\ \alpha_j, & \text{otherwise} \end{cases} \quad (1 \le j \le m) \\
\overline{x_i} &\mapsto \begin{cases} \top_i, & \text{if } \phi(x_i) = \bot \\ \bot, & \text{otherwise} \end{cases} \\
\overline{x_{i,1}} &\mapsto \omega_1 \\
\overline{x_{i,2}} &\mapsto \omega_2 \\
\overline{x_{i,j}} &\mapsto s(k_j) \quad (3 \le j < i+2) \\
\overline{x_{i,i+3}} &\mapsto \begin{cases} \top_i, & \text{if } \phi(x_i) = \bot \\ \bot, & \text{otherwise} \end{cases} \\
\overline{x_{i,j}} &\mapsto s(k_j) \quad (i+4 \le j \le n+2) \\
\overline{x_{i,n+3}} &\mapsto \omega_3 \\
\overline{x_{i,n+4}} &\mapsto \omega_4 \\
\overline{x_{i,j+n+4}} &\mapsto \begin{cases} \top_i, & \text{if } \overline{x_i} \in C_j \\ \alpha_j, & \text{otherwise} \end{cases} \quad (1 \le j \le m) \\
z_i &\mapsto s(k_{i+3}).
\end{aligned}
$$

To complete the definition of s, we need only add the mappings

$$C_j \mapsto \alpha_j$$

for all $i \le j \le m$. Figure 10 shows a portion of s for a particularly simple formula.

Having defined s, let us now show that it is an st-pseudo-schedule. We note

first that the cycle

$$\mathcal{L}_0 = k_0, k_{0,1}, k_1, \ldots, k_{m+n+4}, k_{m+n+4,0}, k_0$$

is bidirectional. Every vertex $k_{i,j}$ not on $\mathcal{L}_0$ has a bidirectional edge to either k_i or k_j; and whichever vertex cannot be reached directly from $k_{i,j}$ is reachable via $\mathcal{L}_0$. Therefore s is an st-pseudo-schedule over the color gadget.

For each variable gadget $\mathfrak{X}_i$, either the path $k_{i+3}, x_{i,i+3}, x_i$ or $k_{i+3}, \overline{x_{i,i+3}}, \overline{x_i}$ is bidirectional in s, depending on whether $\phi(x_i)$ is true or false, respectively; let us denote the bidirectional path by $\mathcal{L}_i$. Furthermore, for any $x_{i,j}$ ($\overline{x_{i,j}}$) not in $\mathcal{L}_i$, there is bidirectional edge with k_j when it is colored differently than k_i; otherwise it has a bidirectional edge with x_i ($\overline{x_i}$). As $x_i, z_i, \overline{x_i}$ is also bidirectional, this and $\mathcal{L}_i$ provides a means for every vertex in $\mathfrak{X}_i$ to reach any other vertex in the same variable gadget, or via $\mathcal{L}_0$ any vertex in the color gadget and all other variable gadgets.

Finally, the edge between a clause vertex C_j and a literal vertex is bidirectional if the literal is true-colored—but every clause vertex is adjacent to a true-colored literal since ϕ satisfies $\mathcal{F}$. □

It is worth noting in the proof above that how we color $G(\mathcal{F})$ is, in some sense, tightly constrained by the very structure of the graph. This turns out to be an essential factor in proving the converse lemma (and makes the proof less intricate).

Lemma 4.5. *Given some formula $\mathcal{F}$, if there exists a (spanning tree) pseudo-schedule of $G(\mathcal{F})$ in $m + n + 5$ colors, then $\mathcal{F} \in$ 3SAT.*

Proof. Observe that if a vertex v is adjacent only to the two distinct vertices x and y, then any (spanning tree) pseudo-schedule of the graph must assign x and y to different colors—for if not, neither (x, v) nor (y, v) are nonconflicting, and there is no nonconflicting path to v. It follows immediately that the color gadget $\mathfrak{K}_{n,m}$

requires $m+n+5$ colors, and thus so does $G(\mathcal{F})$. Let s be a (st-)pseudo-schedule of $G(\mathcal{F})$ in $m+n+5$ colors; without loss of generality we may assume that the colors of $k_0, \ldots, k_{m+n+4}$ are named as in the proof of Lemma 4.4.

Reasoning as above, it must be that $s(x_i), s(\overline{x_i}) \in \{\bot, \top_i\}$, and further that $s(x_i) \neq s(\overline{x_i})$, for all $1 \leq i \leq n$. Therefore a truth assignment ϕ can be defined by the natural interpretation of the color of each positive literal vertex.

Each clause vertex is reachable by a nonconflicting path only if there is at least one true-colored vertex adjacent to it, which corresponds exactly to the condition of ϕ being a satisfying assignment for $\mathcal{F}$. □

We can now proceed to the completeness proof.

Proof of Theorem 4.2. Given some formula $\mathcal{F}$ with n vertices and m clauses, we can generate $G(\mathcal{F})$ in time polynomial to the length of $\mathcal{F}$, as there are $O((m+n)^2)$ vertices. By combining Lemma 4.4 and Lemma 4.5 we see that $\mathcal{F} \in$ 3SAT iff $(G(\mathcal{F}), m+n+5) \in$ PSCHED. Thus there is a polynomial-time reduction from 3SAT to PSCHED. As 3SAT is NP-Complete [1], PSCHED is NP-Hard. By Lemma 4.1 the language PSCHED is in NP, so we conclude that PSCHED is NP-Complete.

The proof for STPSCHED is identical in form. □

4.2 Hardness of approximation

Theorem 4.2 suggests that the search for optimal polynomial-time algorithms is unlikely to be fruitful, or more precisely, success is too fruitful to be likely. We therefore resort to approximate algorithms, as in §3, but we would like to understand if our algorithms are tight. Unfortunately we are far from understanding this for the pseudo-scheduling problem, but the reduction presented in the previous section immediately rules out the existence of an FPTAS, assuming P $\neq$ NP.

The presentation in this section follows Vazirani's textbook [3].

A problem Π is *strongly NP-Hard* iff every problem in NP can be polynomially reduced to Π in such a way that numbers in the reduced instance are written in unary; in other words, numbers are polynomially bounded by the length of the original instance. 3SAT is vacuously strongly NP-Hard since its natural instance form, the formula, contains no numbers.

PSCHED and STPSCHED do contain numbers, but the reduction from 3SAT described in the previous section produces the number $m + n + 5$, where m is the number of clauses and n the number of variables in the 3SAT instance. This is clearly polynomially (actually linearly) bounded by the 3SAT instance, so PSCHED and STPSCHED are strongly NP-Hard.

Proof of Theorem 4.3. Observe that $k = \widetilde{\chi}_{1,1}$ ($k = \widehat{\chi}_{1,1}$) is bounded by the length of any instance of PSCHED (STPSCHED) written in unary, since the instance describes the graph, and k is trivially no greater than the number of vertices. Garey and Johnson [4] showed that a strongly NP-Hard problem satisfying this condition refuses an FPTAS, unless P = NP. □

List of References

[1] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972.

[2] S. T. McCormick, "Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem," *Mathematical Programming*, vol. 26, pp. 153–171, 1983.

[3] V. V. Vazirani, *Approximation Algorithms*. Springer, 2003.

[4] M. R. Garey and D. S. Johnson, "Strong NP-completeness results: motivation, examples, and implications," *Journal of the ACM*, vol. 25, no. 3, pp. 499–508, 1978.

CHAPTER 5

Pseudo-(h, k)-Labelings

Recall from §2.1 that we defined $L(\tau)$-labelings in a very general way, allowing τ to be a tuple with any number of nonincreasing positive integer parameters. Yet we quickly abandoned this generality and considered only the case $\tau = (1, 1)$. Let us now permit $\tau = (h, k)$; we quickly perceive that the $L(h, k)$-labeling can be relaxed in exactly the same sense as $L(1, 1)$-labeling was relaxed to $\widetilde{L}(1, 1)$-labeling.

Given integers h, k such that $h \geq k \geq 1$ and some labeling function $l : V(G) \to \mathbb{Z}$, an ordered pair $(u, v) \in V(G)^2$ is *nonconflicting* under l iff

- $\{u, v\} \in E(G)$;
- $|l(u) - l(v)| \geq h$; and
- $\forall x \in N_G[v] : x \notin \{u, v\} \implies |l(u) - l(x)| \geq k$.

A directed path from u to v is nonconflicting relative to l iff the pairs that comprise it are nonconflicting in l. Finally, a $\widetilde{L}(h, k)$-*labeling*, or *pseudo-*(h, k)-*labeling*, of G is a labeling l such that there exists a nonconflicting path in l from u to v, and vice-versa, for any connected $u, v \in V(G)$; the associated chromatic number is the *pseudo-*(h, k)-*number*, denoted $\widetilde{\chi}_{h,k}$. (Recall that for a labeling l with image K, in general we say that l uses $(\max K - \min K + 1)$ colors.)

An edge uv is *bidirectional* if (u, v) and (v, u) are nonconflicting, of course. Definitions for the *spanning tree pseudo-*(h, k)-*labeling* and its associated chromatic number $\widehat{\chi}_{h,k}$ are obtained just as in §3.1.

The $L(2, 1)$-labeling is considered a viable model for the radio channel assignment problem [1, 2], so $\widetilde{L}(2, 1)$-labelings have relevance for the broadcast scheduling

problem in radio networks. $L(h,k)$-labelings with $h \geq 3$ or $k \geq 1$ appear to be of purely theoretical interest; Calamoneri [3] surveys known results.

In this chapter we will establish bounds and hardness results on $\widetilde{\chi}_{h,k}$.

Theorem 5.1. *Fix $h \geq k \geq 1$. Given a graph G with degree $\Delta \geq 1$,*

$$h + k(\Delta^* - 1) + 1 \leq \widetilde{\chi}_{h,k}(G) \leq \widehat{\chi}_{h,k}(G) \leq 2\Delta(2k-1) + 2(h - 2k + 1)$$

where Δ^* is the smallest possible degree of a spanning tree of G.

Fiala et al. [4] proved that the decision problem version of $L(h,k)$-labeling is NP-Complete. This is also the case for $\widetilde{L}(h,k)$-labeling. (Formal definitions of PLABEL and STPLABEL are given in §5.2 below.)

Theorem 5.2. *Fix $h \geq k \geq 1$.* $\text{PLABEL}_{h,k}$ *and* $\text{STPLABEL}_{h,k}$ *are NP-Complete.*

We can also rule out the existence of an FPTAS; since the proof is basically identical to the proof of Theorem 4.3, we present the result without further comment.

Theorem 5.3. *Fix $h \geq k \geq 1$. Neither* $\text{PLABEL}_{h,k}$ *nor* $\text{STPSCHED}_{h,k}$ *admit an FPTAS, unless* $\text{P} = \text{NP}$.

5.1 Bounds on $\widetilde{\chi}_{h,k}$

Proof of Theorem 5.1. By the same argument used in the proof of Lemma 2.3, from a pseudo-(h,k)-labeling s we can obtain a directed spanning tree T of G expressing nonconflicting paths. Let v be a vertex such that $\deg_T(v) = \Delta_T$. The smallest number of colors used in $N_T[v]$ occurs when v has the smallest color α in the neighborhood, for any neighbor u of v we have (u,v) nonconflicting, and $s(u) = \alpha + h + ki$ where $0 \leq i \leq \Delta_T - 1$. The lower bound follows from this.

To obtain the upper bound, we need only modify the proof of Theorem 3.5, which calculated the upper bound of the 2Δ algorithm. Observe that a vertex

needs to have color h away from its parent, excluding at most $2h-1$ colors; and furthermore the color must be k away from the as many as $2(\Delta_G - 1)$ vertices at distance two that are checked, excluding at most $2(\Delta_G - 1)(2k-1)$ colors. Summing the number of excluded colors and adding one for the vertex itself yields the upper bound. □

The lower bound is tight: for any $m \geq 1$, take $K_{m,1}$, color the dominant vertex with 0, and color the remaining vertices $h, h+k, \ldots, h+k(m-1)$. I do not know if the upper bound is tight—recall that this is unresolved even for $h = k = 1$.

On a related note, let us observe that all the algorithms presented in §3 will work for $\widetilde{L}(h,k)$-labeling after making the obvious modifications, with the exception of the d-band algorithm. Even there, however, we can simply multiply every palette by h, although this seems wasteful when $h > k$. A more sophisticated choice of palette may be possible in this case.

5.2 Hardness

For fixed integer parameters $h \geq k \geq 1$, let $\text{PLABEL}_{h,k}$ be the set of ordered pairs (G, d) such that

$$(G,d) \in \text{PLABEL}_{h,k} \iff \widetilde{\chi}_{h,k}(G) \leq d.$$

The set can be described as a language of strings by canonical encoding methods. We also define $\text{STPSCHED}_{h,k}$ where

$$(G,d) \in \text{STPSCHED} \iff \widehat{\chi}_{h,k}(G) \leq d.$$

To establish Theorem 5.2, we will proceed somewhat differently than in §4. In that chapter we reduced from 3SAT; here we reduce from NAE3SAT, the language of Boolean formulas in conjunctive normal form with exactly three literals per clause that are satisfiable with a truth assignment that exhibits at least one false

literal per clause. NAE3SAT can be described as a set:

$$\left\{ F; \exists\phi \forall C \in F \exists l_1, l_2 \in C : l_1 = \begin{cases} \overline{x}, & \text{if } \phi(x) = \bot \\ x, & \text{otherwise} \end{cases}, l_2 = \begin{cases} y, & \text{if } \phi(y) = \bot \\ \overline{y}, & \text{otherwise} \end{cases} \right\}$$

where the *formula* F is a set of *clauses* C, each clause being a three-element subset of the *literals* L over *variables* X such that $L = \{x, \overline{x}; x \in X\}$. The function $\phi : X \to \{\bot, \top\}$ is a *truth assignment*.

NAE3SAT was shown to be NP-Complete by Garey and Johnson [5]. Without loss of generality, we may assume that no clause contains both x and $\overline{x}$ for any variable x; and that for any variable x, both x and $\overline{x}$ appear in (different) clauses of F. (Observe that a formula not conforming to these assumptions can be transformed into an equivalent conforming formula in polynomial time.)

Schematically, our proof method will be to show that for every (possibly empty) well-formed formula F, one can efficiently construct a graph G_F such that F admits a satisfying assignment iff G_F has a $\widetilde{L}(h,k)$-labeling in some efficiently-computable number of colors.

5.2.1 Hardness of $\text{PLABEL}_{h,1}$

It is not difficult to show the hardness of PLABEL(h,k) by our constructions if the hardness of PLABEL$(h,1)$ is already established, so we take the latter problem first. From this point forward, we consider a formula F with m clauses $C_1, \ldots, C_m$ and n variables $x_1, \ldots, x_n$, conforming to the assumptions mentioned above.

The *formula graph* G_F is comprised of a *color gadget* $\mathfrak{K}_{h,m,n}$; n *variable gadgets* $\mathfrak{X}_1, \cdots, \mathfrak{X}_n$; and vertices $C_1, \ldots, C_m$ coinciding with the clauses of F. Intuitively, the color gadget forces to minimum number of colors to a certain "useful" level. The variable gadgets connect to the color gadget in such a way that particular colors can be identified with the Boolean values $\top$ and $\bot$, certain vertices can be identified with the literals, and these "literal" vertices are labeled with "Boolean"

colors in a meaningful manner. Finally, each clause vertex is set adjacent to the literals it contains.

Let us first consider the color gadget $\mathfrak{K}_{h,m,n}$. Its vertex set is comprised of $N = 2h^2 + 2h + m + n$ main vertices and $\binom{N}{2}$ intermediate vertices.

$$V(\mathfrak{K}_{h,m,n}) = \{k_i; 0 \leq i < N\} \\ \cup \{k_{i,j}; 0 \leq i < j < N\}.$$

The intermediate vertices can be seen as "straddling" the edges in what would otherwise be a clique over the main vertices.

$$E(\mathfrak{K}_{h,m,n}) = \{\{k_i, k_{i,j}\}, \{k_{i,j}, k_j\}; 0 \leq i < j < N\}.$$

The critical thing to note is that no two main vertices can be assigned the same color in any pseudo-labeling; for if this were the case for two main vertices k_a, k_b $(a < b)$, then there could be no nonconflicting pair terminating at the intermediate vertex $k_{a,b}$.

Now consider the variable gadget $\mathfrak{X}_i$ where $1 \leq i \leq n$. There are two literal vertices, with one intermediate vertex between them, and $N - 2$ intermediate vertices per literal. These last will provide the bridge to the color gadget.

$$V(\mathfrak{X}_i) = \{x_i, \overline{x_i}, z_i\} \\ \cup \{x_{i,j}, \overline{x_{i,j}}; 1 \leq j < N, j \neq N - m - 1\}.$$

The edges of the vertex gadget are also constructed according to the principle of "straddling" edges in order to force color difference.

$$E(\mathfrak{X}_i) = \{\{x_i, z_i\}, \{\overline{x_i}, z_i\}\} \\ \cup \{\{x_{i,j}, x_i\}, \{\overline{x_{i,j}}, \overline{x_i}\}; x_{i,j}, \overline{x_{i,j}} \in V(\mathfrak{X}_i)\}.$$

At this point we are ready to assemble the formula graph itself. The vertices are simply the vertices of the assorted gadgets, plus the clause vertices.

$$V(G_F) = V(\mathfrak{K}_{h,m,n}) \cup \bigcup_{i=1}^{n} V(\mathfrak{X}_i) \cup \{C_1, \ldots, C_m\}.$$

Finally we connect the color gadget to the variable gadgets, and the variable gadgets to the clauses (while, of course, retaining all the edges previously introduced).

$$\begin{aligned} E(G_F) = E(\mathfrak{K}_{h,m,n}) &\cup \bigcup_{i=1}^{n} E(\mathfrak{X}_i) \\ &\cup \bigcup_{i=1}^{n} \{\{x_{i,j}, k_j\}, \{\overline{x_{i,j}}, k_j\}; x_{i,j}, \overline{x_{i,j}} \in V(\mathfrak{X}_i)\} \\ &\cup \bigcup_{i=1}^{m} \{\{l, C_i\}; l \in C_i\}. \end{aligned}$$

The global structure of G_F induces a sufficient condition on the existence of a satisfying assignment.

Lemma 5.4. *Fix $h \geq 1$ and let F be some formula of m clauses over n variables. If there exists a (spanning tree) pseudo-$(h,1)$-labeling of the formula graph G_F in $2h^2 + 2h + m + n$ colors, then $F \in$ NAE3SAT.*

Proof. If $m = n = 0$, the result is vacuously true, so assume not. The color gadget ensures that no $\widetilde{L}(h,1)$-labeling occurs in fewer than $N = 2h^2 + 2h + m + n$ colors. Let s be a (spanning tree) pseudo-$(h,1)$-labeling of G_F in N colors. Since we can name colors anything we want, let $s(k_0)$ be the "false color" $\perp$, and $s(k_{N-m-1})$ be the "true color" $\top$.

There is an intermediate vertex in the variable gadget $\mathfrak{X}_i$ between either literal vertex and every main vertex of the color gadget, save k_0 and k_{N-m-1}, so x_i and $\overline{x_i}$ must be colored either $\perp$ or $\top$. Furthermore, the intermediate vertex between these literals compels them to be colored differently from one another. Therefore we can define a Boolean function ϕ over the variables based on whether each x_i is colored "true" or "false."

Now consider any clause vertex C_j where $1 \leq j \leq m$. In order for a non-conflicting pair terminating at C_j to exist, at least one of the three literal vertices adjacent to C_j must be colored uniquely relative to the other two. Thus ϕ is a not-all-equal satisfying assignment of F. □

Next we establish the converse, at least for nearly every case.

Lemma 5.5. *Fix $h \geq 2$ and let F be some formula of m clauses over n variables. If $F \in$ NAE3SAT, then G_F admits a spanning tree pseudo-$(h, 1)$-labeling in $2h^2 + 2h + m + n$ colors.*

Proof. Let $N = 2h^2+2h+m+n$; we will construct a spanning tree $\widetilde{L}(h, 1)$-labeling $s : V(G_F) \to \{0, \ldots, N-1\}$ based on a not-all-equal satisfying assignment ϕ of F.

We begin with the color gadget $\mathfrak{K}_{h,m,n}$. The main vertices are simple enough:

$$k_i \mapsto i$$

for all $0 \leq i < N$.

From this point forward, we implicitly take colors and subscripts modulo N and let $k_{b,a}$ be an alias for $k_{a,b}$ where $a < b$. (Thus, for instance, $k_{N-1,N} = k_{N-1,0} = k_{0,N-1}$.) For all $0 \leq i < N$ and $1 \leq j \leq h^2$:

$$k_{i,i+j} \mapsto \begin{cases} i + h(h - j + 1), & \text{if } 1 \leq j < h \\ i, & \text{otherwise} \end{cases}.$$

Finally, for all $0 \leq i < j < N$ such that $k_{i,j}$ remains uncolored:

$$k_{i,j} \mapsto \begin{cases} j, & \text{if } i = 0 \\ i, & \text{if } 1 \leq i < h \\ j, & \text{if } N - h - m \leq j < N - m - 1 \\ i, & \text{if } j = N - m - 1 \\ j, & \text{otherwise} \end{cases}$$

where the mapping is defined by the first listed condition that is true. (For example, $k_{0,N-m-1} \mapsto N - m - 1$ because the condition $i = 0$ comes before $j = N - m - 1$.)

We say that a color α is "free" on vertex v when $\alpha \notin s(N_{G_F}[v])$. Note that

- the colors 0 and $N-m-1$ are free on all k_i where $h^2+h \leq i < h^2+h+n$;
- the color h^2+h+j is free on k_i and $k_{N-m-i-1}$ for all $1 \leq i < h$ and $0 \leq j < n$; and
- the color h^2+h+j is free on k_i for all $N-m \leq i < N$ and $0 \leq j < n$.

This will come in handy when we color the vertex gadgets.

Claim. The function s is a spanning tree $\widetilde{L}(h,1)$-labeling of $\mathfrak{K}_{h,m,n}$.

Proof. The following set T of edges, which contains a spanning tree of $\mathfrak{K}_{h,m,n}$, are all bidirectional:

$$\begin{aligned} T = &\{\{k_i, k_{i,i+1}\}; 0 \leq i < N\} \\ &\cup \{\{k_i, k_{i,j}\}; 0 \leq i < j < N, |s(k_i) - s(k_{i,j})| \geq h\} \\ &\cup \{\{k_{i,j}, k_j\}; 0 \leq i < j < N, |s(k_j) - s(k_{i,j})| \geq h\}. \end{aligned}$$

It is not hard to see that pairs terminating in an intermediate vertex are nonconflicting, but this is not at all obvious for pairs terminating in a main vertex, as a main vertex has very many neighbors, all of which must be colored differently than the originating element of the pair.

Fix some main vertex k_i. k_i has, in a manner of speaking, h^2 intermediate vertices $k_{i-h^2,i}, \ldots, k_{i-1,i}$ "to the left" and h^2 intermediate vertices $k_{i,i+1}, \ldots, k_{i,i+h^2}$ "to the right." The remaining intermediate vertices $k_{i,j}$ adjacent to k_i present no potential conflicts at k_i, since they will either be colored i or j.

We partition the intermediate vertices to the left and right as follows:

$$L_2 = \{k_{i-h^2,i}, \ldots, k_{i-h,i}\}$$

$$L_1 = \{k_{i-h+1,i}, \ldots, k_{i-1,i}\}$$

$$R_1 = \{k_{i,i+1}, \ldots, k_{i,i+h-1}\}$$

$$R_2 = \{k_{i,i+h}, \ldots, k_{i,i+h^2}\}$$

and note that these vertices take colors in the intervals

$$[i-h^2, i-h], [i+1, i+h^2-1], [i+2h, i+h^2], [i, i]$$

respectively. N is large enough to ensure that disjoint intervals remain disjoint even modulo N, so we conclude that the only vertices potentially conflicting at k_i are in $L_1 \cup R_1$; ie, they lie within "radius" $h-1$ of k_i.

Let $0 < a, b < h$ and consider the vertices $k_{i-a,i}$ and $k_{i,i+b}$ with colors $i-a+h(h-a+1)$ and $i+h(h-b+1)$, respectively. If these colors were equal (modulo N) then

$$a + h(h-a+1) - h(h-b+1) \equiv 0 \implies h(b-a) \equiv a$$

which is impossible: if $b-a<0$, then N is too large for the congruence, as $-h^2 < h(b-a) \leq -h$; if $a=b$, we have $a=h$; and if $b-a>0$, then $h \leq h(b-a) < h^2$, and once again N is too large for the congruence. We conclude that vertices in $L_1 \cup R_1$ do not cause conflicts at k_i, and since k_i was chosen arbitrarily, the claim is proved. □

An immediate consequence of the claim is the truth of the lemma for the empty formula.

We are now ready to complete the construction of s. The clauses are easy to color:

$$C_i \mapsto N - m + i - 1$$

for all $1 \leq i \leq m$.

For the variable gadget $\mathfrak{X}_i$ where $1 \leq i \leq n$:

$$x_i \mapsto \begin{cases} 0, & \text{if } \phi(x_i) = \bot \\ N-m-1, & \text{otherwise} \end{cases}$$

$$\overline{x_i} \mapsto \begin{cases} N-m-1, & \text{if } \phi(x_i) = \bot \\ 0, & \text{otherwise} \end{cases}$$

$$z_i \mapsto h^2 + i - 1.$$

Here, clearly, we are designating 0 as the "false color" and $N-m-1$ as the "true color." As ϕ is a not-all-equal satisfying assignment, we are guaranteed that for every clause vertex C_q $(1 \leq q \leq m)$ there exists some literal $l \in C_q$ such that (l, C_q) is nonconflicting.

Let $\alpha = h^2 + h + i - 1$. For all $1 \leq j < N$ such that $j \neq N-m-1$:

$$x_{i,j} \mapsto \begin{cases} \begin{cases} \alpha, & \text{if } \phi(x_i) = \bot \\ j, & \text{otherwise} \end{cases}, & \text{if } 1 \leq j < h \\ j, & \text{if } h \leq j < h^2 + i - 1 \\ \alpha, & \text{if } j = h^2 + i - 1 \\ j, & \text{if } h^2 + i \leq j < h^2 + h + i - 1 \\ \begin{cases} N-m-1, & \text{if } \phi(x_i) = \bot \\ 0, & \text{otherwise} \end{cases}, & \text{if } j = h^2 + h + i - 1 \\ j, & \text{if } h^2 + h + i \leq j < N - h - m \\ \begin{cases} j, & \text{if } \phi(x_i) = \bot \\ \alpha, & \text{otherwise} \end{cases}, & \text{if } N - h - m \leq j < N - m - 1 \\ \begin{cases} \alpha, & \text{if } x_i \in C_{j-N+m+1} \\ j, & \text{otherwise} \end{cases}, & \text{otherwise} \end{cases}.$$

Note that if the vertex $x_{i,j}$ is colored j, the edge $\{x_i, x_{i,j}\}$ is bidirectional; if it is colored α, then $\{x_{i,j}, k_j\}$ is bidirectional since α is free on k_j; and for $x_{i,h^2+h+i-1}$, both incident edges are bidirectional. Furthermore, the edge $\{x_i, z_i\}$ is bidirectional, and (C, x_i) is a nonconflicting pair for all clauses C such that $x_i \in C$.

We now color the intermediate vertices adjacent to negative literals. We can use essentially the same approach, with all appropriate "switching" for the negation. Also, we no longer need a special case for $j = h^2 + h + i - 1$, for reasons we

will discuss shortly. So with i, j, α as in the preceding paragraph:

$$\overline{x_{i,j}} \mapsto \begin{cases} \begin{cases} j, & \text{if } \phi(x_i) = \bot \\ \alpha, & \text{otherwise} \end{cases}, & \text{if } 1 \leq j < h \\ j, & \text{if } h \leq j < h^2 + h + i - 1 \\ \begin{cases} 0, & \text{if } \phi(x_i) = \bot \\ N - m - 1, & \text{otherwise} \end{cases}, & \text{if } j = h^2 + h + i - 1 \\ j, & \text{if } h^2 + h + i \leq j < N - h - m \\ \begin{cases} \alpha, & \text{if } \phi(x_i) = \bot \\ j, & \text{otherwise} \end{cases}, & \text{if } N - h - m \leq j < N - m - 1 \\ \begin{cases} \alpha, & \text{if } \overline{x_i} \in C_{j-N+m+1} \\ j, & \text{otherwise} \end{cases}, & \text{otherwise} \end{cases}.$$

Observe that the edge $\{\overline{x_i}, z_i\}$ is bidirectional. The pair $(C, \overline{x_i})$ is nonconflicting for all clauses C such that $\overline{x_i} \in C$.

Let B_i denote the bidirectional edges in $\mathfrak{X}_i$. If $e \in B_i$ is incident on the color gadget, we attach it to the bidirectional spanning tree already described on the color gadget; otherwise add it to a set T_i. Now observe that the bidirectional path $x_i, x_{i,h^2+h+i-1}, k_{h^2+h+i-1}$ "bridges" the spanning tree on the color gadget and T_i; hence we conclude that s is a spanning tree pseudo-(h, k)-labeling over the color and variable gadgets.

To complete the proof, it suffices to show just one bidirectional edge incident on each clause vertex—but this is precisely what the assignment ϕ allows us to claim. □

At this point we have enough to show hardness in the case $h > k = 1$, but in fact we are already so close to the general case that we can proceed directly to the proof.

Proof of Theorem 5.2. The proof of Lemma 4.1 establishes that the problems are in NP for the case $h = k = 1$, and the algorithm given there is easily adapted for the general case.

NP-Hardness of the case $h = k = 1$ was established in §4.1, so consider every other case. Given formula F in m clauses and n variables, consider G_F as defined above. Due to the structure of the color gadget, the only way to color it is to assign a unique color to every vertex, and then multiply each color by k. The number of colors is then

$$k(2h^2 + 2h + m + n) - k + 1 = k(2h^2 + 2h + m + n - 1) + 1.$$

We claim that G_F admits a (spanning tree) pseudo-(h, k)-labeling in $k(2h^2 + 2h + m + n - 1) + 1$ colors iff $F \in$ NAE3SAT. For the forward implication, we argue precisely as in the proof of Lemma 5.4, modifying only the number of colors as per the preceding considerations. For the reverse implication, we simply color G_F as in the proof of Lemma 5.5, then multiply each color by k.

Our claim shows that there is a reduction from NAE3SAT to PLABEL$_{h,k}$ ($STPLABEL_{h,k}$). Furthermore, this reduction is polynomial-time, since G_F has only $O((m + n)^2)$ vertices. (Remember that h, k are treated as fixed constants.) We conclude that PLABEL$_{h,k}$ ($STPLABEL_{h,k}$) is NP-Hard. □

List of References

[1] J. R. Griggs and R. K. Yeh, "Labeling graphs with a condition at distance 2," *SIAM Journal on Discrete Mathematics*, vol. 5, pp. 586–595, 1992.

[2] C. McDiarmid, "Discrete mathematics and radio channel assignment," in *Recent Advances in Algorithms and Combinatorics*, ser. CMS Books in Mathematics, B. A. Reed and C. L. Sales, Eds. Springer-Verlag, 2003, vol. 11, pp. 27–63.

[3] T. Calamoneri, "The $L(h, k)$-labelling problem: A survey and annotated bibliography," *The Computer Journal*, vol. 49, no. 5, pp. 585–606, 2006.

[4] J. Fiala, T. Kloks, and J. Kratochvíl, "Fixed-parameter complexity of λ-labelings," *Discrete Applied Mathematics*, vol. 113, pp. 59–72, 2001.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

CHAPTER 6

Conclusion

Recall from §1 that the study offered to present a new formal framework for explicit scheduling solutions to the broadcast scheduling problem that would both be algorithmically tractable and reduce schedule size. Our investigation has surely demonstrated that the $\widetilde{L}(1,1)$-labeling is such a framework, and we have made substantial progress in the formal analysis of this new type of graph coloring problem. While we can be satisfied that the study is complete in that sense, by no means have we exhausted the subject.

In this, the concluding chapter, we will consider a series of questions that we have either had to leave unresolved or were not able to raise until now.

6.1 Open problems

Problems are presented in the order in which the relevant topics were encountered.

Open Problem 1. *Establish whether the upper bound* 2Δ *on* $\widetilde{\chi}_{1,1}$ *is tight. If not, find a tight upper bound; if so, characterize the graphs that achieve the upper bound.*

As mentioned in §2.3, the only graphs I know that achieve the 2Δ upper bound are K_2—for which the upper bound equals the lower bound—and C_{3k+2} $(k \geq 1)$, although in this case $\Delta+2$ may be what is "really" happening. On the other hand, 2Δ is an intuitively appealing upper bound. I consider this the top open problem, since it represents a chink in the armor of fundamental results.

If 2Δ is a tight bound, we would like to establish a kind of "Brooks' Theorem" for it by characterizing the graphs that hit the bound. Theorem 2.8 demonstrates

that such a class, if it exists, is "small."

Open Problem 2. *Characterize the graphs G such that $\widetilde{\chi}_{1,1} > \Delta_G + 1$.*

Refer to Table 2 on p. 29: the empirical data suggests that graphs with pseudo-$(1,1)$-number above $\Delta + 1$ are "rare." I even find them hard to construct deliberately. A characterization of such graphs would be immensely useful; merely having meaningful necessary or sufficient conditions would also be desirable, particularly if they related to graph classes frequently studied as network models (scale-free, unit disk, etc).

Open Problem 3. *Let $p = p(n)$ be "sufficiently far" from zero or one (eg, $0 < p < 1$ is a constant). Establish whether*

$$\widetilde{\chi}_{1,1}(G) < \Delta_G$$

for almost every random graph $G = G(n,p)$.

Referring again to Table 2, the empirical data suggests that the expectation μ of $(\widetilde{\chi}_{1,1} - \Delta)$ goes to $-\infty$ as $n \to \infty$. If this is true, and furthermore the variation can be shown to be $o(\mu^2)$, by Chebyshev's inequality we can establish the desired result. Indeed, we would have $\widetilde{\chi}_{1,1} - \Delta \sim \mu$ almost always, a very powerful result, provided we can say anything specific about μ.

Open Problem 4. *Establish whether $\widetilde{\chi}_{1,1}(G) = \widehat{\chi}_{1,1}(G)$ for all graphs G. If not, characterize the graphs for which this holds and investigate the preceding open problems relative to $\widehat{\chi}_{1,1}$.*

In Figure 5 on p. 24 we saw that not every pseudo-schedule is spanning tree, but I do not know any graph for which $\widetilde{\chi}_{1,1} \neq \widehat{\chi}_{1,1}$. If indeed $\widetilde{\chi}_{1,1} = \widehat{\chi}_{1,1}$, we would gain a powerful proof tool, since the additional structure of st-pseudo-schedules makes them easier to investigate. We would also be spared a lot of work, since a

possibly strict inequality would compel us to "re-ask" every question about $\widetilde{\chi}_{1,1}$ in terms of $\widehat{\chi}_{1,1}$.

Open Problem 5. *Given a graph G, establish which classes of spanning subgraphs H admit H-pseudo-schedules in $O(\Delta_G)$ colors, and find efficient (approximate) algorithms for H-pseudo-schedules. Furthermore, investigate the same problem where G and/or H are directed graphs.*

With T a spanning tree of G, a T-pseudo-schedule is "fragile" in the sense that the removal of any edge in T may disconnect G relative to the set of nonconflicting paths, even though G is not actually disconnected. This is, of course, an inherent trade-off in the move from strict scheduling to pseudo-scheduling, but nevertheless we may wish to specify spanning subgraphs H that are more resilient (eg, k-connected), particularly for applications in unreliable networks.

Observe that the greedy algorithm (§3.2) works without modification on any spanning subgraph H and even retains the same bounds (with Δ_H in place of Δ_T). The upper bound $O(\Delta_G \Delta_H) = O(\Delta_G^2)$ is undesirable, though, albeit for some H unavoidable in principle (eg, trivially, $H = G$). We would therefore like to identify for which classes of spanning subgraphs we can obtain results asymptotically similar to those for spanning trees.

Finally, we would like to generalize to directed graphs.

Open Problem 6. *Refactor the d-band algorithm for the interference graph.*

Given a graph $G = (V, E)$, we can form an *interference graph* $G_I = (V, E \cup I)$ given a set of directed *interference edges* I; $(x, y) \in I$ is interpreted to mean that x can induce a transmission conflict at y (but not vice-versa, unless $(y, x) \in I$ as well). We then say that (u, v) is nonconflicting under coloring s iff $uv \in E$, $s(u) \neq s(v)$, and u is colored differently from every in-neighbor of v in G_I distinct

from u. A pseudo-schedule is then defined in the normal way, but note that we have effectively required nonconflicting paths to proceed over E, while additionally having to avoid conflicts induced by I.

The interference graph is a more realistic model for broadcast networks, although the fact that one cannot normally assume that communication is possible over interference edges means that the discovery of the interference graph is itself a difficult problem. The protocol RID of Zhou et al. [1] solves the problem by varying transmit power and letting I be more or less synonymous with links realized in a high-power mode but not under normal power.

Assuming the interference graph is discoverable, we would like to refactor the d-band algorithm (§3.4) for such graphs, since we assessed this algorithm to be a strong protocol candidate. Supposing G has an r-rooted tree T, note that the d-band algorithm already works without modification when $|\operatorname{dist}_T(x,r) - \operatorname{dist}_T(y,r)| \neq 1$ for all $(x,y) \in I$, provided d is sufficiently large, since the algorithm ignores all such edges anyway. Thus the easiest modifications would involve choosing r and/or T such that this condition is realized, although I do not know that this is always possible. If indeed not, changes to the algorithm proper are probably unavoidable.

Open Problem 7. *Establish fixed-parameter hardness results for* PSCHED *and* STPSCHED*, and hardness results for particular graph classes.*

For any (cubic) graph G, it is NP-Complete to determine $\chi_{1,1}(G) \leq 4$ [2]. Similarly, we would like to show that there exists some constant c such that it is NP-Complete to determine if the (st-)pseudo-$(1,1)$-number of a graph is more than c. It is reasonable to suspect that $c = 3, 4$.

It is also hard to determine the $(1,1)$-number of many graph classes, including unit disk graphs and planar graphs [2]. We would like to resolve similar questions

for PSCHED (STPSCHED), particularly relative to classes that model networks.

Open Problem 8. *Establish tighter inapproximability results for* $\widetilde{\chi}_{1,1}$ *and* $\widehat{\chi}_{1,1}$.

The discovery the PCP Theorem, which characterizes NP in terms of the class of decision problems with probabilistically checkable proofs, known as PCP, marked a revolutionary advance in the study of inapproximability; see §29 of [3]. Our inapproximability result, Theorem 4.3, is a very weak finding based on techniques that predate the PCP Theorem. Using a modern approach, we would like to derive tighter inapproximability results; I suspect that a proof that $\widetilde{\chi}_{1,1}$ ($\widehat{\chi}_{1,1}$) cannot be approximated within any constant factor may not be too difficult.

Open Problem 9. *Investigate the preceding open problems relative to pseudo-*(h, k)*-labeling.*

The most important case is $h = 2, k = 1$ due to the practical implications for radio channel assignment.

List of References

[1] G. Zhou, T. He, J. A. Stankovic, and R. Abdelzaher, "RID: radio interference detection in wireless sensor networks," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, March 2005, pp. 891–901.

[2] T. Calamoneri, "The $L(h, k)$-labelling problem: A survey and annotated bibliography," *The Computer Journal*, vol. 49, no. 5, pp. 585–606, 2006.

[3] V. V. Vazirani, *Approximation Algorithms*. Springer, 2003.

BIBLIOGRAPHY

Ahn, G., Miluzzo, E., Campbell, A. T., Hong, S., and Curomo, F., "Funneling-MAC: A localized, sink-oriented MAC for boosting fidelity in sensor networks," in *Fourth ACM Conference on Embedded Network Sensor Systems (SenSys 2006)*, November 2006, pp. 293–306.

Alon, N. and Spencer, J. H., *The Probabilistic Method*, 3rd ed., ser. Wiley-Interscience Series in Discrete Mathematics and Optimization. John H. Wiley & Sons, 2008.

Bollobás, B., *Random Graphs*, 2nd ed., ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

Broersma, H., Fomin, F. V., Golovach, P. A., and Woeginger, G. J., "Backbone colorings for graphs: tree and path backbones," *Journal of Graph Theory*, vol. 55, no. 2, pp. 137–152, 2007.

Calamoneri, T., "The $L(h, k)$-labelling problem: A survey and annotated bibliography," *The Computer Journal*, vol. 49, no. 5, pp. 585–606, 2006.

Chipara, O., Lu, C., and Stankovic, J., "Dynamic conflict-free query scheduling for wireless sensor networks," in *Fourteenth IEEE International Conference on Network Protocols (ICNP '06)*, November 2006, pp. 321–331.

Chlamtac, I., Faragó, A., and Zhang, H., "Time-spread multiple access protocols for multihop mobile radio networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 804–812, 1997.

Cowen, L., Goddard, W., and Jesurum, C. E., "Defective coloring revisited," *Journal of Graph Theory*, vol. 24, no. 3, pp. 205–219, 1997.

DiPippo, L., Tucker, D., Fay-Wolfe, V., Bryan, K. L., Ren, T., Day, W., Murphy, M., Henry, T., and Joseph, S., "Energy-efficient MAC for broadcast problems in wireless sensor networks," in *Third International Conference on Networked Sensing Systems*, June 2006.

Dressler, F., "A study of self-organization mechanisms in ad hoc and sensor networks," *Computer Communications*, vol. 31, no. 13, pp. 3018–3029, 2008.

Dysktra, R. L., Hewett, J. E., and Thompson, Jr., W. A., "Events which are almost independent," *The Annals of Statistics*, vol. 1, no. 3, pp. 674–681, 1973.

Fiala, J., Kloks, T., and Kratochvíl, J., "Fixed-parameter complexity of λ-labelings," *Discrete Applied Mathematics*, vol. 113, pp. 59–72, 2001.

Fürer, M. and Raghavachari, B., "Approximating the minimum-degree Steiner tree within one of optimal," *Journal of Algorithms*, vol. 17, no. 3, pp. 409–423, 1994.

Garey, M. R. and Johnson, D. S., "Strong NP-completeness results: motivation, examples, and implications," *Journal of the ACM*, vol. 25, no. 3, pp. 499–508, 1978.

Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

Griggs, J. R. and Yeh, R. K., "Labeling graphs with a condition at distance 2," *SIAM Journal on Discrete Mathematics*, vol. 5, pp. 586–595, 1992.

Grötschel, M., Lovász, L., and Schrijver, A., "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1980.

IEEE 802.11 Working Group, *ANSI/IEEE Std 802.11*, 1999th ed., IEEE.

Karp, R. M., "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, Miller, R. E. and Thatcher, J. W., Eds. Plenum Press, 1972.

Levin, B., "Simple improvements on Cornfield's approximation to the mean of a noncentral hypergeometric random variable," *Biometrika*, vol. 71, no. 3, pp. 630–632, 1984.

McCormick, S. T., "Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem," *Mathematical Programming*, vol. 26, pp. 153–171, 1983.

McDiarmid, C., "Discrete mathematics and radio channel assignment," in *Recent Advances in Algorithms and Combinatorics*, ser. CMS Books in Mathematics, Reed, B. A. and Sales, C. L., Eds. Springer-Verlag, 2003, vol. 11, pp. 27–63.

Molloy, M. and Reed, B., *Graph Colouring and the Probabilistic Method*, ser. Algorithms and Combinatorics. Springer, 2002.

Nowakowski, R. and Winkler, P., "Vertex-to-vertex pursuit in a graph," *Discrete Mathematics*, vol. 43, pp. 235–239, 1983.

Polastre, J., Hill, J., and Culler, D., "Versatile low power media access for wireless sensor networks," in *Second ACM Conference on Embedded Network Sensor Systems (SenSys 2004)*, November 2004, pp. 95–107.

Prisner, E., "Convergence of iterated clique graphs," *Discrete Mathematics*, vol. 103, pp. 199–207, 1992.

Ramanathan, S. and Lloyd, E. L., "Scheduling algorithms for multihop radio networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 2, pp. 166–177, 1993.

Ren, T., "Graph coloring algorithms for TDMA scheduling in wireless sensor networks," Ph.D. dissertation, University of Rhode Island, 2007.

Rhee, I., Warrier, A., Aia, M., and Min, J., "Z-MAC: a hybrid MAC for wireless sensor networks," in *Third ACM Conference on Embedded Network Sensor Systems (SenSys 2005)*, November 2005, pp. 90–101.

Rhee, I., Warrier, A., Min, J., and Xu, L., "DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks," in *Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)*, May 2006, pp. 190–201.

Riordan, O. and Selby, A., "The maximum degree of a random graph," *Combinatorics, Probability and Computing*, vol. 9, pp. 549–572, 2000.

Scheinerman, E. R. and Ullman, D. H., *Fractional Graph Theory.* Self-published, 2008. [Online]. Available: http://www.ams.jhu.edu/~ers/fgt

Sundararaman, B., Buy, U., and Kshemkalyani, A. D., "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.

Vazirani, V. V., *Approximation Algorithms.* Springer, 2003.

Zhou, G., He, T., Stankovic, J. A., and Abdelzaher, R., "RID: radio interference detection in wireless sensor networks," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, March 2005, pp. 891–901.