On the Complexity of Link Scheduling in Multi-hop Radio Networks

Errol L. Lloyd[1] and S. Ramanathan[2]

Dept. of Computer and Information Sciences
University of Delaware, USA.

Abstract

The problem of efficiently scheduling the links of a multi-hop radio network is studied using a graph theoretic representation. It is shown that link scheduling is NP-complete even when restricted to planar graphs. In contrast to the general case for which no good approximation algorithms are known, we describe a constant times optimal algorithm for the case of planar graphs. For the general case, this algorithm is shown to produce a schedule which is bounded by a length proportional to the graph *thickness* times optimal. The behaviour of the algorithm is experimentally studied in comparison with existing methods. These results show that the algorithm presented here improves (in terms of worst-case performance and average performance) upon previous methods.

1 Introduction

A *radio network* is a network of processors which communicate using broadcast radio. A radio communication network is *single-hop* if all of the stations are within transmission range of each other. Otherwise, it is a *multi-hop* network. If a receiving station in a radio network is within range of two or more simultaneously transmitting stations, a transmission *collision* occurs at the site of the receiver. A *schedule* is an assignment of time 'slots' for transmissions in such a way that the transmissions assigned to the same slot do not collide.

The problems associated with the construction of an efficient schedule have received considerable attention over the past decade [Ari84, EWB87, CL85, Pro89]. The constraints imposed on the simultaneity of two or more station transmissions depend on the capabilities of the network stations and on the actual signalling technique used[BG87, EWB87]. For most networks, including the ones we consider in this paper, the following hold.
1. At any given time, a station may either transmit or receive (but not both).
2. A station tuned to a particular transmitter may not be within range of another transmitting station.

The problem that we consider in this paper is that of *link scheduling*, wherein the links between nodes are assigned time slots for *activation.* That is, the intended receiver of a transmission is scheduled to receive at the same time the transmitter is scheduled to transmit.

A standard representation of a radio network is in terms of a directed graph G=(V,A). Here, V is a set of *vertices* denoting the stations in the radio network, and A is a set of directed edges between vertices such that for any two distinct vertices u,v $\in$ V, edge (u$\rightarrow$v) $\in$ A if and only if v can receive u's transmission. Note that we do not make an apriori assumption about the edges of the corresponding graph being bidirectional.[3] That is, (u$\rightarrow$v) $\in$ A does not necessarily imply (v$\rightarrow$u) $\in$ A.

A natural interpretation of scheduling in this context is as one of *coloring* the corresponding graph G=(V,A). Link scheduling corresponds to *coloring* the edges of the graph such that any pair of directed edges (a$\rightarrow$b) $\in$ A and (c$\rightarrow$d) $\in$ A may be colored the same if and only if

[1]elloyd@udel.edu
[2]ramanath@udel.edu
[3]In some networks, extraneous noise or deliberate jamming at the site of one station[EWB87] may cause one-way connections.

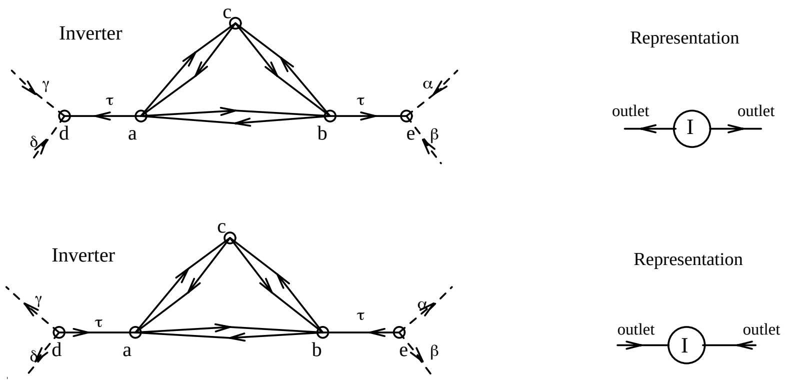

Figure 1: The Inverters

1. a, b, c, d are all mutually distinct, and
2. (a→d) $\notin$ A and (c→b) $\notin$ A.

In subsequent sections, we will interchangeably use the terminology of *prn coloring*[4] and link scheduling to refer to the abovementioned problem. It has been shown that the *optimal* prn coloring of a directed graph is NP-complete[EGMT84, Ari84, RL92b]. Thus, only approximate (ie, good, but not necessarily optimal) solutions are likely to exist. All of the currently existing algorithms[CL85] have a worst-case performance bounded by O(ρ^2) (where ρ is the maximum vertex degree). Finding a good approximation algorithm seems to be a difficult algorithm. Indeed, in [RL92a], we provide evidence that indicates a close relationship between prn coloring and traditional vertex coloring[GJ79] for which it is widely believed that no good algorithms exist.

In this paper, we consider a restricted version of prn-coloring. We show that the prn-coloring is NP-complete *even when restricted to planar graphs.* We then give an approximation algorithm that provides, for planar graphs, a worst-case performance bound of O(ρ). That is, the algorithm uses no more than a constant times optimal number of colors[5]. Our analysis of the algorithm for the *general* case shows that the performance is no worse than O($\theta\rho$) where θ is the *thickness* of the graph. The thickness is the minimum number of planar graphs whose union is the graph, and may be percieved informally as a measure of the "nearness to planarity" of the graph. Since the thickness is typically several orders of magnitude lesser than the maximum degree, this improves on the best existing performance of O(ρ^2). We also analyze our algorithm experimentally and compare its performance with other algorithms.

The remainder of this paper is organized as follows. In section 2, we describe a proof of the NP-completeness of prn coloring (link scheduling). In section 3, we give an approximation algorithm for prn coloring and analyze it for arbitrary and planar graphs. Section 4 discusses the experimental model we use and the results.

2 Link Scheduling of Planar Networks is NP-complete

To study the complexity of the Link Scheduling problem, we consider the equivalent coloring problem **7-(planar) prn-coloring** stated below.

INSTANCE : A (planar) directed graph G=(V,A).
QUESTION : Is G 7-prn colorable, ie, is there an assignment f : A → {1,2...}, such that for any a→b, c→d $\in$ A, f(a→b) = f(c→d) only if a,b,c,d are all distinct and a→d $\notin$ A and c→b $\notin$ A.

Theorem. The 7-planar prn coloring problem is NP-complete.

Proof. We prove the theorem using a reduction from the problem of 7-distance-2 coloring given below,

[4]The problem is historically associated with Packet Radio Networks (PRNs)
[5]note that the optimum, for any graph of degree ρ, is at least ρ

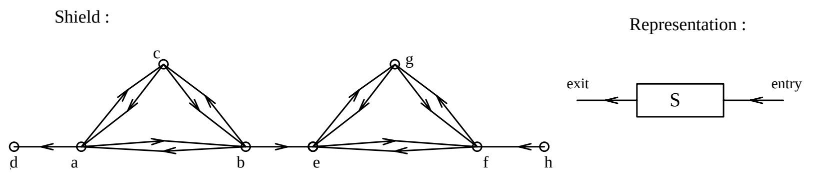

Figure 2: The Shield

which is known to be NP-complete. [RL92a, GJ79].

INSTANCE : A (planar) undirected graph G=(V,E).
QUESTION : Is G 7-distance-2 colorable, ie, is there an assignment f : V $\rightarrow$ {1,2,3..}, such that for u,v $\in$ V, f(u) = f(v) only if there does not exist a path of length less than or equal to 2, from u to v.

We use the technique of component design [GJ79] in our reduction. We give below, three lemmas on the components illustrated in Figures 1, 2, 3. We state the first two lemmas without proof due to space restrictions, and refer the reader to [RL92b] for details.

Lemma 2.1. In any 7-prn-coloring of the inverter shown in Figure[6] 1(a), color(a$\rightarrow$d) = color(b$\rightarrow$e); In any 7-coloring of the *inverter* shown in Figure 1(b), color(a$\rightarrow$d) = color(b$\rightarrow$e).

Lemma 2.2. In any 7-coloring of the *shield* shown in Figure[7] 2, color(a$\rightarrow$d) = color(h$\rightarrow$f).

Lemma 2.3. In any 7-coloring of the *crossing* shown in Figure 3, (color(w$\rightarrow$p) = color(y$\rightarrow$k)) $\neq$ (color(x$\rightarrow$f) = color(a$\rightarrow$u) = color(c$\rightarrow$v)).

Proof. Assume without loss of generality that edges f$\rightarrow$r, x$\rightarrow$f, d$\rightarrow$f, f$\rightarrow$d, f$\rightarrow$e, e$\rightarrow$f and d$\rightarrow$e are colored using colors 1,2,3,4,5,6 and 7 respectively. Note that these edges mutually interfere and therefore, *must* be colored differently. Then, d$\rightarrow$h *must* be colored 1. Thus, the only possible color available for b$\rightarrow$e is 2. By lemma 2.1 applied to the inverter delineated by vertices a,b,c, we have color(a$\rightarrow$u) = color(c$\rightarrow$v) = color(b$\rightarrow$e) = 2 = color(x$\rightarrow$f). Furthermore, color(x$\rightarrow$f) $\neq$ color(w$\rightarrow$p) since x$\rightarrow$f is adjacent to f$\rightarrow$r and color(f$\rightarrow$r) = color(w$\rightarrow$p). □

We represent the above mentioned components using 'symbols' as shown alongside the respective figures. We observe that, in any 7-prn-coloring
1. The outlets of an inverter must be colored the same.
2. The entry and exit of the shield must be colored the same.
3. The entry, exit and xmit outlets of the crossing must be colored the same. The rcv and meet outlets of the crossing must also be colored the same, but this color must be different from the colors of the entry, exit and xmit.

Given an instance G of distance-2 coloring, we construct an instance G$'$ of prn-coloring as follows. For every node u $\in$ V, we construct a ring R$_u$ of ρ (maximum degree) crossings, (each adjacent vertex of u being 'assigned' one crossing) connected to each other by the 'entry' and 'exit' outlets as shown in Figure 4. We denote the crossing corresponding to vertex v as C_u^v. For u,v such that (u,v) $\in$ E, we place a shield between xmit(C_u^v) and recv(C_v^u) so that the outlets of the shield are identified with xmit(C_u^v) and recv(C_v^u) as shown in Figure 4. Further, we create a new node B$_u$ in the 'center' of each R$_u$. We let each of the meet($C_u^{v_i}$), be incident on B$_u$ for i=2,4,6. For i=1,3,5, we let it be identified with an outlet of an inverter, and let the other outlet of the same inverter be incident on B$_u$, as shown in Figure 4. Note that since G is 7 colorable, its maximum degree is at most 6 and hence i $\leq$ 6.

We now show that G is 7-distance-2 colorable if and only if G$'$ is 7-prn colorable. We first consider the 'if' part and then the 'only if' part.

1. G$'$ is 7 prn-colorable $\Rightarrow$ G is 7-distance-2 colorable. Let f$'$ be a legal 7-prn coloring of G$'$. For all u $\in$ G, we let f(u) be the color of the 'entry' edges of the crossings (all the entry (and exit) edges must have the same color, by lemma 2.3) in R$_u$. The proof is by contradiction. We claim that f is a legal

[6]Ignore the dashed edges and the greek symbols in the figure for the moment
[7]Ignore the dashed edges and greek symbols in the figure for the moment

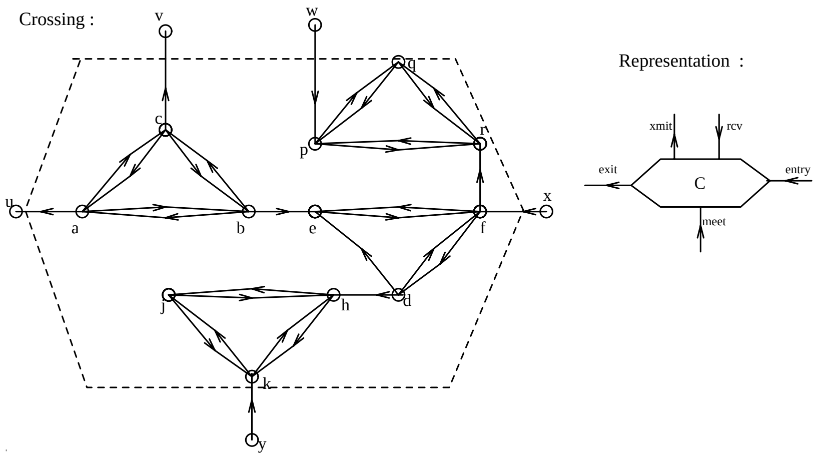

Figure 3: The Crossing

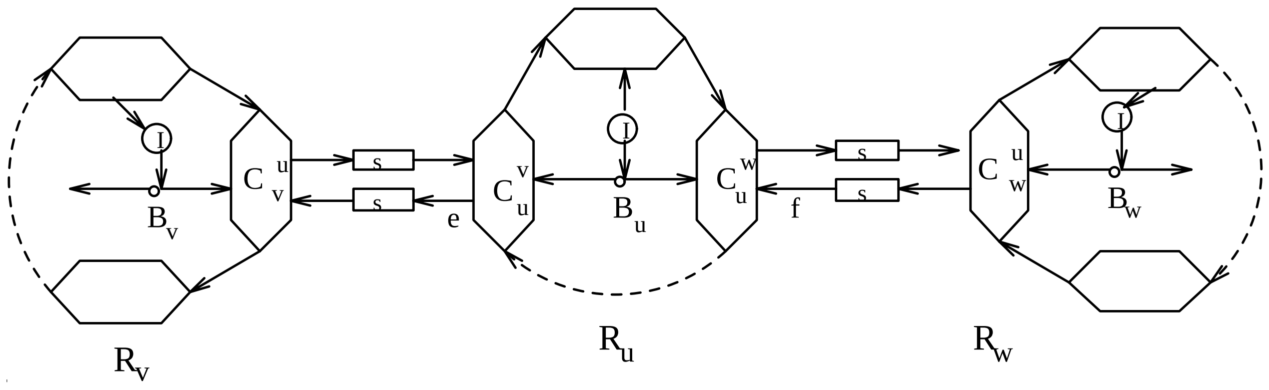

Figure 4: Construction for reduction of distance-2 coloring to prn coloring. Every vertex x is replaced by a ring R$_x$ of crossings. Adjacent nodes rings are linked through a shield, by identifying the relevant outlets. For instance, edge e is both the xmit of the crossing and the outlet of the shield. The edge f is both the rcv of the crossing and the outlet of the shield.

coloring of G. For, if it were not, one of the following cases must be true.
Case 1: Two adjacent vertices get the same color. Suppose (u,v) $\in$ E and f(u) = f(v). This would imply, by our construction, that $f'(\text{entry}(C_u^v)) = f'(\text{entry}(C_v^u))$. But $f'(\text{entry}(C_v^u)) = f'(\text{xmit}(C_v^u))$ (by lemma 2.3) = $f'(\text{rcv}(C_u^v))$ (by construction and lemma 2.2. See Figure 4). Thus, $f'(\text{entry}(C_u^v)) = f'(\text{rcv}(C_u^v))$. But by lemma 2.3, the 'entry' and 'rcv' edges of any crossing must be colored differently.
Case 2: Two vertices adjacent to another vertex get the same color. Let u,v,w be vertices such that (v,u) $\in$ E, (w,u) $\in$ E and f(v) = f(w). This would imply, by our construction, that $f'(\text{entry}(C_w^u)) = f'(\text{entry}(C_v^u))$. By a reasoning similar to that in case 1 above, we can see that $f'(\text{rcv}(C_u^v)) = f'(\text{rcv}(C_v^w))$. By lemma 2.3, the 'rcv' and 'meet' of any crossing have to be colored the same. Thus, $f'(\text{meet}(C_u^v)) = f'(\text{meet}(C_u^w))$. However, it is easy to see from our construction that none of the 'meet' edges can be colored the same, since they are either adjacent, or are forced by an inverter to have the same colors as that of adjacent edges.

Thus, our assumption that f' is a legal coloring is contradicted in each of the two cases above. Hence, f must be a legal coloring.

2. G is 7-distance-2 colorable $\Rightarrow$ G' is 7-prn colorable.

We describe a procedure to construct a 7-prn coloring of G' from a 7-distance-2 coloring of G. Let f be a legal 7-distance-2 coloring of G. We begin by coloring the outlets of each of the crossings in each of the rings. In particular, we let, for all u $\in$ V, $f'(\text{entry}(C_u^{v_i})) = f'(\text{exit}(C_u^{v_i})) = f'(\text{xmit}(C_u^{v_i})) = f(u)$, and $f'(\text{rcv}(C_u^{v_i})) = f'(\text{meet}(C_u^{v_i})) = f'(v_i)$, for i = 1,2,3 .. ρ.

The coloring thus far is clearly legal. In the coloring of the remainder of the edges of G', some situations recur frequently. We first study these *scenarios* in detail, and then use those results to complete the proof. These scenarios involve the coloring of a component under the influence of already colored *alien* edges, ie, edges not part of the component, but having an effect on the colors assignable to the edges that *are* part of the component.

Scenario 1. Let I be an inverter, the outlets of which are adjacent to two alien edges each, shown in Figure 1 by dashed edges. Furthermore, let the two alien edges and the outlets be the only edges colored (with colors shown in greek letters). Then, we can 7-prn color the remainder of the edges of the inverter.
Coloring Scheme. We describe the scheme only for the inverter of Figure 1(a). The proof for the other inverter is similar and is omitted here.

The colors are assigned to the remainder of the edges in the following order.
1. Assign b$\rightarrow$a, b$\rightarrow$c colors from {1..7} - {α, β, τ}.
2. Assign a$\rightarrow$c, a$\rightarrow$b colors from {1..7} - {γ, δ, τ, f'(b$\rightarrow$a), f'(b$\rightarrow$c)}.
3. Assign c$\rightarrow$a, c$\rightarrow$b colors from {1..7} - {τ, f'(b$\rightarrow$a), f'(b$\rightarrow$c), f'(a$\rightarrow$c), f'(a$\rightarrow$b)}.

Clearly, at each step, the number of colors available is at least two, so that the procedure may be completed successfully. It is easy to verify that it is a legal coloring.

Scenario 2. Let S be a shield, the outlets of which are adjacent to two alien edges each, shown in Figure 2 by dashed edges. Furthermore, let the two alien edges and the outlets be the only edges colored, as shown in Figure 2. Then, we can 7-prn color the remainder of the edges of the shield.
Coloring Scheme. Color the inverter subgraph delineated by vertices a,b,c using the scheme for scenario 1. Apply this once again to the remainder of the graph to get a legal coloring.

We note that alien edges in incident on vertices d,e in the direction *opposite* to those shown will not influence the coloring of the inverter; a similar fact holds for the shield. An easy extension of these scenarios is to the case when there are *fewer* than the abovementioned number of colored alien edges. Clearly, the same coloring schema suffices.

We are now ready to describe the coloring of G'. We first color all of the crossings in G'. To color a particular crossing, we divide it into four pieces P_1, P_2, P_3, P_4, as shown delineated in Figure 2 and color in that order. With the coloring so far as shown (without loss of generality) in Figure 2, the coloring of the edges in P_1 is done using the following scheme.
1. Assign colors to e$\rightarrow$f, d$\rightarrow$f from {1..7} - {α, β, ρ, σ}.
2. Assign colors to f$\rightarrow$e, f$\rightarrow$d, d$\rightarrow$e from {1..7} - {α, β, f'(e$\rightarrow$f), f'(d$\rightarrow$f)}

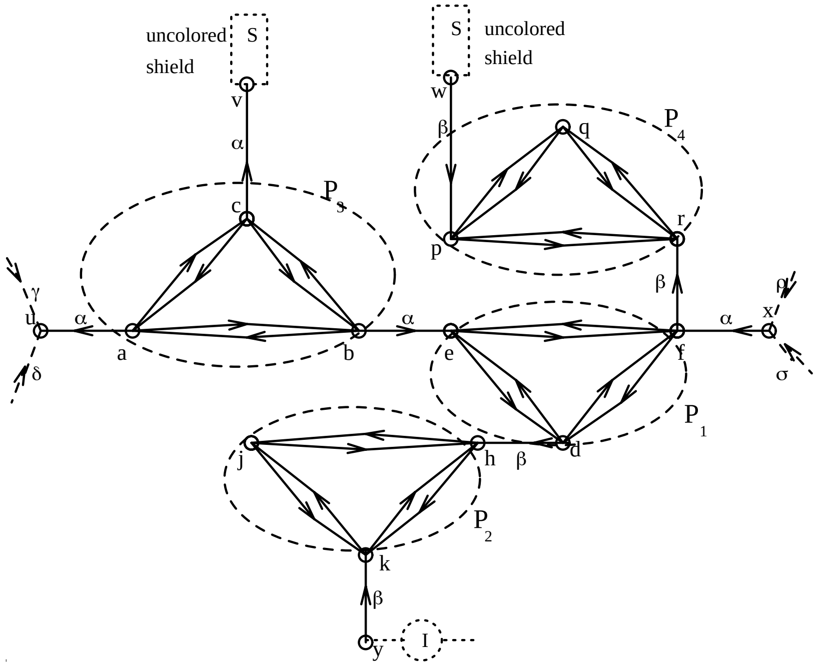

Figure 5: Division of crossing into subcomponents for coloring

Clearly, there are enough colors available at each step and the above coloring is legal. Now, we color P_2. We note that there can be at most two alien edges outgoing from vertex y. This is because, by our construction, $y \rightarrow k$ is either an outlet of an inverter, or $y = B_u$. In the former case, the fact is trivially true. In the latter case, recall that there are exactly three outgoing edges incident on B_u, and $y \rightarrow k$ is one of them. Hence, there are at most two alien outgoing edges incident. Scenario 1 may thus be applied here and the coloring done using the corresponding coloring scheme. To color P_3, we first ignore edge $c \rightarrow v$ and color using the scheme for scenario 1. We then simply note that the addition of $c \rightarrow v$ and the color thereof preserves legality of the coloring. Scenario 1 is applicable to P_4 also, and therefore those edges can be legally colored.

Lastly, we color the shields in between the Rings. The coloring of these shields corresponds to scenario 2, and the coloring may be done as described there, to complete the 7-prn coloring of G'. $\square$

3 Approximate Algorithm for Link Scheduling

In the previous section, we showed that for a planar graph G = (V,A), finding an optimal link schedule is NP-complete. The main result of this section is a new algorithm for link scheduling. The performance guarantee for this algorithm is O(1) for planar graphs and $O(\theta)$ for arbitrary graphs of thickness θ. In what follows, we begin with a description of the algorithm, followed by an analysis of its performance.

3.1 The Algorithm

We first assign a unique *label* to each of the vertices in G. The labelling is performed in a progressive fashion - after the labelling of each vertex, the next vertex to be labelled will be the one having the least total number of neighbors *in the subgraph formed by unlabelled vertices*. Then, we color the edges of the graph by considering the vertices in decreasing order of the labels. For each vertex x, we take each incoming/outgoing edges of x and assign, in a greedy fashion a color to it. That is, we assign the lowest numbered color that can be assigned without causing a conflict with previously colored edges. We note that the ordering of the vertices is crucial to bounding the worst case performance of the algorithm.

The algorithm **Label** follows.

Algorithm Label

input : A directed graph G=(V,A)
output : An assignment of 'labels' L : V $\rightarrow$ {1,2,3,...}

```
1.  let counter ← 1
2.  while there are unlabelled vertices in G do
3.  begin
4.      Find the vertex u with the minimum total
        number of neighbors
5.      set L(u) ← counter
6.      increment counter
7.      delete u (and the edges incident on u)
        from G
8.  end while
```

The actual link scheduling algorithm is as follows.

Algorithm LinkSchedule

input : A planar directed graph G = (V,A)
output : An assignment of colors(slots) c : A $\rightarrow$ {1,2,..}

```
1.  label V
2.  for j= highest label down to lowest label do
3.      Let v be the (only) vertex labelled j.
4.      for each of the edges 'e' incident on v, do
5.        if e is incoming into v
6.        then Init Unavailable color set
          $\mu \leftarrow \phi$
7.              $\mu \leftarrow \mu \cup \{c(v,x) : (v,x) \in A_i\}$
8.              $\mu \leftarrow \mu \cup \{c(u,y) : (u,y) \in A_i\} \{c(y,u) : (y,u) \in A_i\}$
9.              $\mu \leftarrow \mu \cup \{c(a,b) : (a,b) \in A_i,$
                $(u,b) \in A-A_i\}$
10.             $\mu \leftarrow \mu \cup \{c(e,f) : (e,f) \in A_i,$
                $(e,v) \in A-A_i\}$
11.             Let c(u,v) be the first color $r \notin \mu$
12.       else {e is outgoing from v and processed symmetrically}
```

3.2 Analysis

It is easy to see that the algorithm yields a legal coloring. In this section, we analyze the performance bounds and time complexity of LinkSchedule. It is assumed that the input to LinkSchedule is a directed graph G = (V,A) of thickness θ and maximum degree ρ. An important fact that we shall use but do not prove is given below. We give a proof of this in [RL92a].

Fact 1. A graph of thickness θ has at least one vertex whose degree is at most 6θ - 1.

Notice that for the special case of planar graphs ($\theta = 1$), the above implies that a planar graph has a vertex of degree at most five, a well-known fundamental property of planar graphs [BM76]. We need the following lemma before we can examine the performance of LinkSchedule.

Lemma 3.1. In the 'labelling' of G, every vertex v has at most 6θ - 1 neighbors labelled larger than v.

Proof. Since G is of thickness θ, every subgraph of G is of thickness θ. In particular, the subgraph G^j considered in the jth iteration of the loop in Algorithm **label** is of thickness θ. From Fact 1, it is clear that G^j

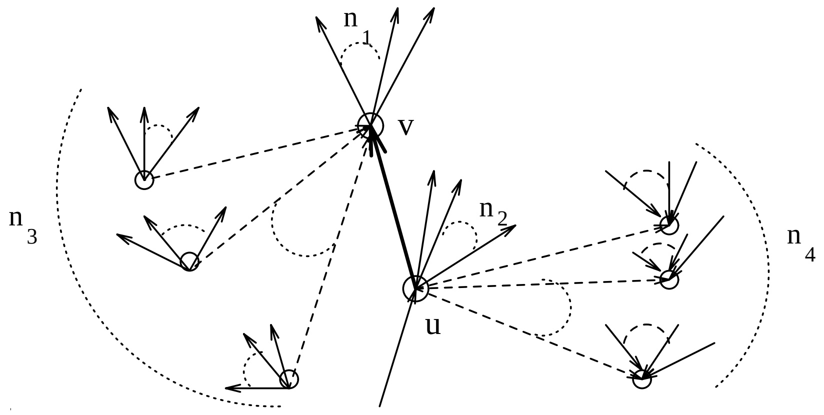

Figure 6: **Neighborhood of edge to be colored in LinkSchedule**

has at least one vertex with at most 6θ - 1 neighbors. Note also that all of the vertices in G^j are unlabelled. It follows that u, the vertex labelled in this iteration has at most five neighbors. Since all of these neighbors are assigned colors in later iterations and 'counter' is incremented in each iteration, the lemma follows. $\Box$

Performance : We show that the algorithm uses no more than $O(\theta\rho)$ colors to color an arbitrary graph. Let u$\rightarrow$v be the edge considered in line 5. Let n_1, n_2, n_3, n_4 be the number of colors added to the set of unavailable colors in lines 7, 8, 9 and 10 respectively. It suffices to show that $n_1 + n_2 + n_3 + n_4$ is $O(\theta\rho)$. It is easy to see from Figure 6 that $n_1 \leq \rho$ - 1, $n_2 \leq \rho$ - 1.

We show that n_3 is $O(\rho)$ by carefully examining interference edges contributing to n_3. To do this, we let m be the label on vertex v. Then, the interference edges $x_i \rightarrow$v contributing to n_3 can be divided into two sets: one set for all of the x_i labelled >m (type 1) and one set for the x_i labelled <m (type 2). Let n_{31} be the number of colored edges incident on those x_i labelled >m and n_{32} the number of colored edges incident on those x_i labelled <m. By lemma 3.1, there are at most 6θ-1 neighbors of v labelled greater than v, and hence there are at most 6θ-1 edges of type 1. Thus, n_{31} is at most $(6\theta\text{-}1)\rho$. On the other hand there can be as many as ρ edges of type 2. Consider one such edge, say $x_1 \rightarrow$v. Of the edges (x_1, y_i) outgoing from x_1, those with y_i labelled less than the label of x_1 are as yet uncolored since their label is <m, and the coloring is performed in decreasing order of labels. Of the edges with y_i labelled greater than the label of x_1, there can be at most 6θ (by lemma 3.1 and the fact that u is labelled >m). Hence, n_{32} is at most $(6\theta)\rho$. Thus, $n_3 = n_{31} + n_{32}$ is $O(\theta\rho)$. By a similar argument on the relative labels of v and the out-neighbors of u, n_4 is easily shown to be $O(\theta\rho)$. The reader is referred to [RL92a] for details.

Previous algorithms for link scheduling all have a bound of $O(\rho^2)$ (a formal analysis is done in [CL85]). Intuitively, it seems that the thickness is much smaller for a graph, in comparison with the maximum degree. Our experiments (see Figure 7, columns 3 and 4) indicate that in practice, the thickness is *several* orders of magnitude less than the maximum degree. Thus, the algorithm LinkSchedule improves significantly on the performance of previous algorithms for the same purpose.

It is now easy to see that the performance of the algorithm, when restricted to planar graphs ($\theta = 1$) is $O(\rho)$. The *optimal* number of colors required to color the graph, clearly, is at least ρ, since there exist ρ adjacent edges. It follows that the performance *guarantee* of the algorithm is O(1). In other words, we have given an algorithm whose performance is no worse than a constant times optimum. By generally accepted notions [CLR90], this is a *good* approximation.

Running Time : Let e and v denote the number of edges and the number of vertices respectively in the input graph G = (V,E). Finding a vertex of minimum total number of neighbors, deleting it and updating the total neighbor count for each of its neighbors can be done in a total time of O(e+vlogv) using Fibonacci Heaps[FT87]. Thus, Algorithm **Label** runs in time O(e + vlogv). Also, it is easy to see that the for-loop in line 2 of LinkSchedule runs in time $O(v\theta\rho)$. Thus, the overall running time of LinkSchedule is O(e + vlogv + $v\theta\rho$).

4 Experimental results : Performance on Average

We now discuss the performance of the algorithm given in section 3.1 in 'practice'. Our investigation involves the construction of a realistic model of broadcast networks and testing the implementation of the algorithm with a large number of 'random' networks generated using the model. The model used by us is discussed below and the experimental results are tabulated following that.

4.1 The Model

In this subsection, we discuss the generation of radio networks that we use for experimentally studying the performance of our algorithms and existing methods.

As noted above, graphs are a natural representation of networks. Thus, in our experimental studies, our goal is to generate a number of 'random' graphs which can then be provided as test inputs to the various scheduling/coloring algorithms. In order to generate graphs that model radio networks as closely as possible, we incorporate two steps. First, we generate the 'geography' of the network. This is done by generating a random position for each of the stations independently, and attributing to each of them a (possibly predetermined) transmission range. Next we convert this into a graph by computing the inter-station distance and placing a pair of directed edges between two vertices if and only if the inter-station distance is less than the range. The experiments described in this paper are restricted to uniformly homogeneous ranges and bidirectional connections. Thus, in our experiments we use two parameters to describe a network (and the corresponding graph):

1. The number of stations S in the network. These stations are randomly placed within a rectangular grid.
2. The transmission range R of each of the stations. We assume that the range of all of the stations is the same, though the model can easily be modified to accomodate variable ranges.

In the tables appearing in section 4.2, the results for each pair of (S,R) values, are obtained by averaging over thirty random graphs generated with those values.

4.2 Experimental Results

In this section we provide an experimental analysis of the performance of LinkSchedule in comparison with exisiting heuristics for the problem of link scheduling. We begin with brief descriptions of the heuristics that were studied.

1. The random choice (RanCho) algorithm. This is a straightforward algorithm in which an edge is chosen at random and colored with the first available, non-conflicting color. Most of the algorithms described in previous works are essentially this method.
2. The maximum degree clique first (MCF) algorithm. In this method we take a *maximal* mutually conflicting clique of edges around the maximum degree vertex first, color it, and then progressively do the same for the remainder of the graph. This method is based on the intuitive notion that it is better to color the more 'crowded' areas first. Heuristics with this philosophy were first examined in [MMI72] for vertex coloring and were found to do quite well.
3. The algorithm LinkSchedule (LSch). This is an implementation of the algorithm LinkSchedule of section 3.1.

The performance of these heuristics (number of slots used for scheduling) is shown in Figure 7. Each entry is averaged over thirty different random graphs with the same parameters. As seen in that table, LinkSchedule generally performs the best, followed by the maximal clique first algorithm and lastly by the

Networks		Graph Params		Colors (slots) used		
Nodes	Range	MaxD	Tkns	RanCho	MCF	LSch
200	20	10	1	19	19	19
200	30	16	2	47	47	46
200	40	24	2	88	87	44
200	50	34	3	157	156	149
400	20	16	2	52	52	51
400	30	29	3	116	113	111
400	40	42	4	226	222	212
400	50	60	5	440	436	409

Figure 7: **Comparative performance of link scheduling algorithms**

commonly used Random Choice algorithm. On average, the LinkSchedule algorithm uses roughly 6 % less slots than the Random Choice algorithm. The difference in performance widens as we tend toward higher ranges and a higher population (higher densities). Note that LinkSchedule does significantly better for a population of 400 nodes each with a range of 50. In the context of the fact that schedules are set up once and used repeatedly many times over, even a small reduction in the number of slots used is worthwhile since it is amplified by the amount of time the schedule is in operation.

Finally, we note that we have conducted a great many more experiments than those described here. These additional experiments considered both a wider variety of algorithms and additional values of S and R. It is notable that in no circumstance did any algorithm have an average performance better than that of LinkSchedule. The *high consistency* of these results certainly gives enhanced credibility to the value of LinkSchedule.

References

[Ari84] E. Arikan. Some complexity results about packet radio networks. *IEEE Transactions on Inform. Theory, vol IT-30*, pages 910–918, Jul 1984.

[BG87] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall Inc., 1987.

[BM76] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* American Elsevier Publishing Co. Inc., New York, 1976.

[CL85] I. Chlamtac and S. Lerner. A link allocation protocol for mobile multi-hop radio networks. In *Proc. Globecom*, Dec. 1985.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill Book Co., 1990.

[EGMT84] S. Even, O. Goldreich, S. Moran, and P. Tong. On the np-completeness of certain network testing problems. *Networks, vol. 14*, pages 1–24, 1984.

[EWB87] A. Ephremedis, J.E. Wieselthier, and D.J. Baker. A design concept for reliable mobile radio networks with frequency hopping signalling. In *Proceedings of the IEEE vol 75, No. 1*, pages 56–73, Jan 1987.

[FT87] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM vol. 34, no. 3*, pages 596–615, 1987.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability : A guide to the theory of NP-completeness.* W. H. Freeman, San Fransicisco, 1979.

[MMI72] D. W. Matula, G. Marble, and J. F. Issacson. Graph coloring algorithms. in Graph Theory and Computing, Academic Press NY, 1972.

[Pro89] C.G. Prohazka. Decoupling link scheduling constraints in multi-hop packet radio networks. *IEEE transactions on Computers, vol. 38, no. 3*, pages 455–458, Mar 1989.

[RL92a] S. Ramanathan and E. L. Lloyd. An algorithmic study of certain broadcast network problems. Technical Report 92-19, Dept. of Computer Science, University of Delaware, 1992.

[RL92b] S. Ramanathan and E. L. Lloyd. Complexity of certain graph coloring problems with applications to radio networks. Technical Report 92-18, Dept. of Computer Science, University of Delaware, 1992.