

# SDP-based Approach for Channel Assignment in Wireless Networks

Hieu Dinh\*

Yoo-Ah Kim\*

Seungjoon Lee†

Min-ho Shin†

## Abstract

In wireless networks, due to the broadcast property of the medium, nearby links may interfere with each other and cannot be used simultaneously over the same wireless channel. One way to overcome this limitation is to assign different channels available in the system to those links. Given a graph  $G = (V, E)$ , our goal is to assign channels to links so that we can minimize the total number of conflicts while satisfying constraints on the number of channels that can be used. For a pair of links, they are said to be conflicting if they are within interference range of each other and the channels assigned to them are the same. We consider heterogeneous networks where each node can have a different constraint  $C_v$  on the number of wireless cards, which limits the number of channels that edges incident to  $v$  can use. In addition, we have the constraint on the number of channels that can be used in the network (denoted as  $C_G$ ). We present a semidefinite programming (SDP) formulation for the problem, which provides a lowerbound on the optimal solution. We develop two rounding algorithms based on the optimal solution to SDP. In addition, we present two other simple channel assignment heuristics and conduct experimental evaluations of these algorithms. Our algorithms can be used for general interference models including the two hop interference model and the protocol model.

## 1 Introduction

In wireless networks, due to the broadcast property of the medium, nearby links may interfere with each other and cannot be used simultaneously over the same wireless channel. Consider the example shown in Figure 1. When node  $A$  uses a link to  $B$ , the message will be broadcast to all the neighbors of  $A$  and therefore, the other two links cannot be used over the same channel. One way to overcome this limitation is to utilize independent channels (that can be used without interference) available in the system. If all links use the same channel in the example shown in Figure 1, only one pair of nodes can communicate with each other at a time. However, if there are three channels available in the network and each node is equipped with two wireless interface cards (so each can use two channels), then we can assign a distinct channel to each link and there is no conflicts among links in this channel assignment.

We consider the following CHANNEL ASSIGNMENT problem. We are given a graph  $G = (V, E)$ , and constraints on the number of wireless cards  $C_v$  for all  $v$ , which limit the number of channels that edges incident to  $v$  can use. In addition, we have the constraint on the number of channels available in the network (denoted as  $C_G$ ). For a pair of edges, they are said to be *conflicting* if they are within interference range of each other and the channels assigned to them are the same. Our goal is to assign channels to links so that we can minimize the total number of conflicts while satisfying constraints on the number of channels that can be used. We focus on the two-hop interference model where two edges within two hop distance can interfere with each other. (See Section 2.1 for the detailed description of interference models.)

The problem is NP-hard even for the homogeneous cases ( $C_v = k$  for all  $v$ ) by a reduction from STRONG EDGE COLORING PROBLEM (also called DISTANCE-2 EDGE COLORING) where the goal is to color edges with minimum number of colors so that any two edges within distance two have distinct colors. The STRONG EDGE

---

\*Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269 . E-mail : hdinh@engr.uconn.edu, ykim@engr.uconn.edu.

†Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail: slee@cs.umd.edu, mhshin@cs.umd.edu.

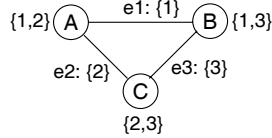


Figure 1: In this network, each node has two wireless interface cards (thus can use two independent channels) and three channels are available in total. We can assign a distinct channel to each link as shown above. With the above channel assignment, there is no conflict among edges.

COLORING problem has been studied extensively in the literature [1, 2, 3]. In our problem, coloring need not be proper (two edges within two hop distance are allowed to use the same color) but the goal is to minimize the number of such conflicts. In addition, each node has its local color constraint, which limits the number of colors that can be used by the edges incident to the node. For example, if a node has two wireless cards ( $C_v = 2$ ), the node can choose two colors and edges incident to the node should use only those two colors. We can show that it is NP-hard even for the case where  $C_v = 1$  or 2.

Many schemes have been proposed to exploit multiple channels for performance improvement in wireless networks. In Multi-radio Unification Protocol (MUP) [4], each node statically assigns a channel to each interface card, and when a node needs to transmit a packet, it checks the channel condition and uses the channel with the best condition at that time. In this protocol, a node with  $C_v$  interface cards, uses channels only from  $\{1, 2, \dots, C_v\}$  even when there are more available channels in the network. We presented [5] heuristics for channel assignment in which the goal is to develop distributed algorithms which utilize multiple channels to improve the network performance while maintaining the connectivity of the network. To make sure that the network is connected, the heuristics first find a spanning subgraph and assign channels so that no edges in the spanning subgraph are dropped. The performance of the heuristics is evaluated via  $ns-2$  simulations. Berrett et al. [2] give sequential and distributed approximation algorithms for channel assignment problem, using STRONG EDGE COLORING. Their goal is to produce a conflict-free assignment using the minimum number of channels or to maximize the number of edges colored when the limited number of channels are given. A number of papers [6, 7, 8] consider the joint problem of channel assignment and routing, and propose centralized algorithms that assign channels to links and find routing paths.

We have recently developed [9] algorithms for channel assignment to minimize the number of conflicts for particular classes of networks, and provide theoretical analysis and approximation factors. The analysis was given only in one-hop interference model (two edges incident to a vertex interfere with each other) and for restricted cases such as homogeneous networks ( $C_v = k$  for all  $v$ ) and semi-heterogenous networks ( $C_v = 1$  or  $k$ ).

**Contributions.** In this paper, we consider the channel assignment problem in general settings. We consider heterogenous networks where each node has different  $C_v$ . We first present a semidefinite programming (SDP) formulation of the problem, which provides a lowerbound on the optimal solution. The solution to SDP can be solved optimally in polynomial time. We develop two rounding algorithms based on the optimal solution to SDP. In addition, we present two other simple channel assignment heuristics and conduct experimental evaluations of these algorithms. We focus on the two hop interference model but our algorithms can be extended to general interference models including the protocol model.

## 2 Models and Definitions

### 2.1 Network and Interference Model

We represent a network as an undirected graph  $G = (V, E)$ , where  $V$  is a set of nodes, and  $E$  a set of links. Let

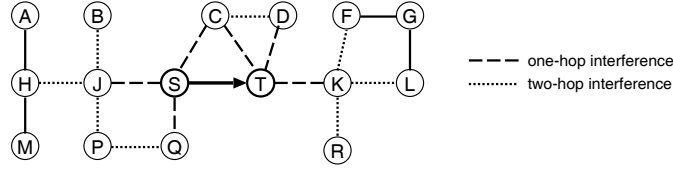


Figure 2: Illustration of one-hop and two-hop interference when the wireless link from  $S$  to  $T$  is used.

$C_G$  be the number of channels available in the system and  $C_v$  be the number of network interface cards at node  $v$ . Without loss of generality, we assume  $\forall v, C_v \leq C_G$ . We assume that each network interface can operate only on one channel at a given time and there is no interference between independent channels. Therefore, a node with  $C_v$  interface cards can use  $C_v$  independent channels without interference.

Consider the following two simplified interference models — one hop interference model and two hop interference model. A node can use a single channel to communicate with only one neighbor node at a time, and we represent such constraint using the *one-hop* interference model [10]. Therefore, an edge  $e$  interferes with the following set of edges:  $I(e) = \{e' | e' \text{ has a common endpoint } v \text{ with } e\}$ . We assume that  $e \notin I(e)$ . For example, in Figure 2, when node  $S$  sends a packet to node  $T$ , all neighboring nodes of  $S$  and  $T$  (e.g.,  $C, D, J, K$ , and  $Q$ ) cannot send data packets to  $S$  and  $T$  (shown in dashed lines). This one-hop interference model is simple and uses only one-hop neighborhood information, but it sometimes does not capture the real wireless communication environments. For example, a transmission from  $J$  to  $B$  may collide with the transmission from  $S$  and  $T$  due to the broadcast property of the medium.

In the *two-hop* interference model [11], in addition to the constraint in the one-hop model, edge  $e$  cannot be used if there exists an ongoing transmission over an edge  $e'$  where either endpoint of  $e$  is a neighbor of either endpoint of  $e'$ . That is,  $I(e) = \{e' | e' \text{ has a common endpoint or are adjacent to a common edge } e''\}$ . For example, in Figure 2,  $I(e = (S, T)) = \{e' | e' \text{ is incident to } C, D, J, S, T, K, \text{ and } Q\}$  (shown in dotted or dashed lines). However, any edge in solid line (e.g., from  $G$  to  $L$ ) can be used together with  $e = (S, T)$ . In this paper we focus on channel assignment in the two-hop interference model. Even though we present the results based on the two-hop interference model, our schemes can be extended to more general interference models including the protocol model [12].

## 2.2 Problem Definition

We are given a graph  $G = (V, E)$  where  $v \in V$  is a node in a wireless network and an edge  $e = (u, v) \in E$  represents a communication link between  $u$  and  $v$ . Each node  $v$  can use  $C_v$  different channels and the total number of channels that can be used in the network is  $C_G$ . More formally, let  $E(v)$  be the edges incident to  $v$  and  $c(e)$  be the color assigned to  $e$ . Then  $|\bigcup_{e \in E(v)} c(e)| \leq C_v$  and  $|\bigcup_e c(e)| \leq C_G$ . We focus on the two-hop interference model discussed in Section 2.1. Therefore, for an edge  $e$ , its interference set  $I(e)$  is defined as  $\{e' | e \text{ and } e' \text{ have a common endpoint or are adjacent to a common edge } e'', e' \neq e\}$ . Also we define  $I(e_1, e_2)$  to be 1 if  $e_2 \in I(e_1)$  and 0 otherwise. A pair of edges  $e_1$  and  $e_2$  is said to be conflicting if the two edges use the same color and  $I(e_1, e_2) = 1$ . Let us define the *conflict number* ( $CF_e(A)$ ) of an edge  $e \in E$  in a channel assignment  $A$  to be the number of other edges that conflicts with  $e$ . In other words, for an edge  $e$ ,  $CF_e(A)$  is the number of edges in  $I(e)$  that use the same channel as  $e$  in assignment  $A$ . Our goal is to find a channel assignment that minimizes the total number of conflicts, which is defined as

$$CF_G(A) = \frac{1}{2} \sum_{e \in E} CF_e(A). \quad (2.1)$$

In the remainder of this paper, we mean *channels* by *colors* and use edge coloring and channel assignment,

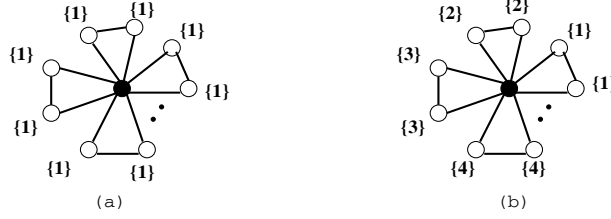


Figure 3: Black node has  $k$  interface cards and white nodes have only one interface card. (a) the solution in NAIVE ALGORITHM. all links use channel 1. (b) optimal solution. each of  $k$  channels is used by the same number of links.

interchangeably. We also use conflict and interference interchangeably.

### 3 Channel Assignment Algorithms

In this section, we describe several channel assignment algorithms. We first describe a simple heuristic and its limitation, and present more sophisticated algorithms to overcome the limitation.

#### 3.1 A Naive Algorithm

We consider the following simple algorithm for channel assignment (See Algorithm 1). For each node  $v$  with  $C_v$ , we allow the node to use channels  $\{1, \dots, C_v\}$ . Therefore, an edge  $e = (u, v)$  can use a channel from  $\{1, \dots, \min(C_u, C_v)\}$ . An edge chooses its channel greedily as follows. Let  $n(e, i)$  be the number of edges using color  $i$  in  $B(e)$  where  $B(e)$  includes all edges in  $I(e)$  that have chosen their colors *before*  $e$ . Then  $e$  chooses a channel with minimum value of  $n(e, i)$  from  $\{1, \dots, \min(C_u, C_v)\}$ . That is, whenever we choose a color for an edge, we try to minimize the number of conflicts that are created.

---

#### Algorithm 1 NAIVE ALGORITHM

---

```

for each edge  $e = (u, v)$  do
     $S = \{1, \dots, \min(C_u, C_v)\}$ .
    assign color  $c \in S$  with min value of  $n(e, c)$  to  $e$ .
end for

```

---

In homogeneous networks where  $C_v = k$  for all  $v$ , it has been shown that the number of conflicts given by the algorithm in one hop interference model is at most  $(1 - \frac{1}{k})|E|$  more than the optimal [9]. Moreover, the approximation ratio is (asymptotically) best possible unless  $P = NP$ . In more general settings, however, such an approximation bound does not hold.

Note that in this algorithm the total number of different colors used may be much smaller than  $C_G$ . Consider the example shown in Figure 3. The black node has  $k$  interface cards and all other nodes have only one interface. In this case, NAIVE ALGORITHM ends up using the same color for all edges whereas in the optimal solution, we can distribute edges evenly for each of  $k$  color.

In the following section, we describe GREEDY ALGORITHM, where we try to utilize all  $C_G$  colors available in the network. The difficulties when we use more colors are in that it may not be possible to maintain the connectivity of graph when each edge chooses its color only with local information.

#### 3.2 Greedy Algorithm

To utilize more channels, we allow each edge to choose any channel from  $\{1, \dots, C_G\}$  as long as it satisfies the

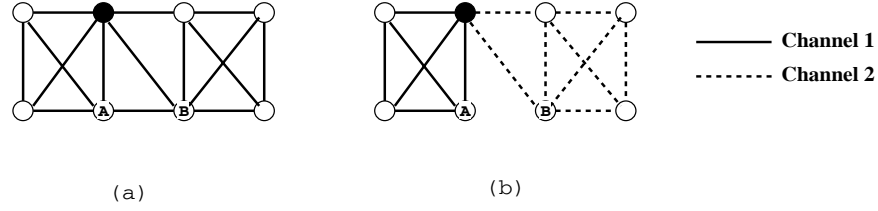


Figure 4: Black node has 2 interface cards and white nodes have only one interface card. (a) all links have to use the same channel if we keep all edges. (b) If we assign channel 1 to node  $A$  and channel 2 to node  $B$ , then we can reduce the number of conflicts significantly. Node  $A$  and  $B$  cannot communicate with each other directly as there is no common channel. However, the two nodes can send/receive the message via other path from  $A$  and  $B$ .

color constraints  $C_v$ . The algorithm works as follows (See Algorithm 2). When we decide a color for an edge, we first find the set of colors that can be used for the edge. Let  $|S_v|$  be the number of colors that any edge in  $E(v)$  is already using. If  $|S_u| = C_u$  and  $|S_v| = C_v$ , then edge  $e = (u, v)$  need to choose a color from  $S = S_u \cap S_v$ . Otherwise if either of sets is tight, we choose the set as  $S$ . If we can add a new color to both of sets,  $e$  can use any  $C_G$  colors. Recall that  $n(e, i)$  denotes the number of edges which are already using color  $i$  in  $I(e)$ . As in NAIVE ALGORITHM, we choose color  $c$  with minimum  $n(e, c)$  for edge  $e$ . In case that  $S = \emptyset$ , we drop edge  $e$  (i.e., node  $u$  and  $v$  cannot communicate with each other directly).

---

**Algorithm 2** GREEDY ALGORITHM

---

```

for each edge  $e = (u, v)$  do
  if  $|S_v| = C_v$  and  $|S_u| = C_u$  then
     $S = S_u \cap S_v$ .
  else if  $|S_v| = C_v$  and  $|S_u| < C_u$  then
     $S = S_v$ .
  else if  $|S_v| < C_v$  and  $|S_u| = C_u$  then
     $S = S_u$ .
  else
     $S = \{1, \dots, C_G\}$ .
  end if
  if  $S = \emptyset$  then
    drop edge  $e$ .
  else
    let  $c \in S$  be the color with  $\min n(e, c)$ .
    assign color  $c$  to edge  $e$ .
    add  $c$  to  $S_v$  and  $S_u$  if not included.
  end if
end for

```

---

While the greedy approach may reduce the number of conflicts by utilizing more channels, the biggest drawback of this algorithm is that it drops a significant number of edges. Since each edge chooses its color greedily, some edges may end up being dropped when color sets of two endpoints are disjoint and an additional color cannot be added due to color constraints. In fact, there are instances for which we can reduce the total

number of conflicts significantly by dropping just a few edges. Consider the example shown in Figure 4. If we keep all the links, then the same channel need to be used by all links, but by just dropping the link from  $A$  and  $B$ , we can reduce the number of conflicts significantly.

On the other hand, if an edge is dropped, then the pair of nodes will have to use another route to communicate with each other, thus increasing the traffic. Therefore, dropping too many edges is not desirable, and more importantly, the network should remain connected after dropping edges. As we can see in the experiments (Section 4), GREEDY ALGORITHM drops 10 – 25% of edges in heterogenous cases and sometimes the network is disconnected. This is because each edge chooses its color without global information. In the following section, we introduce an SDP-based algorithm, where an edge chooses its color based on the optimal solution to SDP relaxation of the channel assignment problem.

### 3.3 SDP-based Rounding Algorithm

We first formulate the problem as a vector programming and obtain an optimal solution by converting it to a semidefinite programming (SDP). We obtain a channel assignment by rounding the solution to SDP. The rounding algorithm is greedy in nature but as the solution is guided by an optimal solution to SDP, we found that the rounded solution improves the performance while having less edges dropped than GREEDY ALGORITHM.

**Semidefinite Programming Formulation:** Consider the following vector programming (VP), which we can convert to SDP and obtain an optimal solution in polynomial time. We have an  $m$ -dimensional unit vector  $X_e$  for each edge  $e$  and  $Y_v$  for each node  $v$  ( $m \leq |V| + |E|$ ). Let  $C_G = k$ .

**VP:**

$$\min \sum_{e_1, e_2} \frac{1}{k} ( (k-1)X_{e_1} \cdot X_{e_2} + 1 ) \quad \text{for } I(e_1, e_2) = 1 \quad (3.2)$$

$$|X_e| = 1 \quad (3.3)$$

$$|Y_v| = 1 \quad (3.4)$$

$$X_{e_1} \cdot X_{e_2} \geq \frac{-1}{k-1} \quad \text{for } I(e_1, e_2) = 1 \quad (3.5)$$

$$Y_v \cdot X_e \geq T_v \quad \text{for } v, e \text{ where } C_v < k \text{ and } e \in E(v) \quad (3.6)$$

where  $T_v = \sqrt{\frac{1}{C_v} \cdot \frac{k-C_v}{k-1}}$ .

We can relate a solution of VP to a channel assignment as follows. Consider  $k$  unit length vectors in  $m$ -dimensional space such that for any pair of vectors  $v_i$  and  $v_j$ , the dot product of the vectors is  $-\frac{1}{k-1}$ . (It has been shown that  $-\frac{1}{k-1}$  is the minimum possible value of the maximum of the dot products of  $k$  vectors [13, 14].) Given an optimal channel assignment of the problem, we can map each channel to a vector  $v_i$ .  $X_e$  takes the vector that corresponds to the channel of edge  $e$ .

Constraints (3.6) ensure that edges in  $E(v)$  can have only  $C_v$  colors. Let  $S_v = \{X_1, X_2, \dots, X_{C_v}\}$  be the vectors corresponding to channels used by edges in  $E(v)$ . Let  $Y_v = \frac{\sum_{i=1}^{C_v} X_i}{|\sum_{i=1}^{C_v} X_i|}$ . In other words,  $Y_v$  is the vector which has the same distance to all vectors in  $S_v$ . Then

$$Y_v \cdot X_i = X_i \cdot \frac{\sum_{i=1}^{C_v} X_i}{|\sum_{i=1}^{C_v} X_i|} = \frac{1 - \frac{C_v-1}{k-1}}{|\sum_{i=1}^{C_v} X_i|} = \frac{1}{|\sum_{i=1}^{C_v} X_i|} \cdot \frac{k - C_v}{k - 1}.$$

Since  $|\sum_{i=1}^{C_v} X_i| = \sqrt{\sum_i \sum_j X_i \cdot X_j} = \sqrt{C_v - C_v(C_v - 1)/(k - 1)}$ , we have Constraints (3.6). For example, let  $C_G = 3$  and  $C_v = 2$ . A feasible channel assignment will correspond to choosing three vectors such that the dot product of any pair is  $-\frac{1}{2}$ . Edges in  $E(v)$  for node  $v$  can use only two out of those three vectors since the dot product of  $Y_v$  and  $X_e$  should be no greater than  $\frac{1}{2}$ .

The objective function is exactly the same as the number of conflicts in the given channel assignment since if  $X_{e_1} = X_{e_2}$  ( $e_1$  and  $e_2$  have the same color), it contributes one to the objective function, and 0 otherwise. Thus the optimal solution of the VP gives a lower bound of the optimal solution for the channel assignment problem.

The above VP can be converted to a semidefinite programming (SDP) and solved in polynomial time (within any desired precision) [15, 16, 17, 18, 19]. We use the optimal solution of SDP to compare the performance of the channel assignment algorithms in Section 4. Given a solution for the SDP, we can find the solution to the corresponding VP, using incomplete Cholesky decomposition [20].

---

**Algorithm 3** SDP-COLORSET ALGORITHM

---

```

solve VP and obtain  $\{X_e, Y_v\}$ .
randomly select  $k$  vectors (Let  $R = \{r_1, r_2, \dots, r_k\}$  be the chosen vectors).
for each vertex  $v$  do
    include  $C_v$  vectors with maximum  $r_i \cdot Y_v$  in  $S_v$ .
end for
for each edge  $e$  do
     $S = S_u \cap S_v$ .
    if  $S = \emptyset$  then
        drop edge  $e$ .
    else
        choose the vector with maximum value of  $r_i \cdot X_e$  from  $S$ .
        assign color  $r_i$  to edge  $e$ .
    end if
end for

```

---

**Rounding Algorithms:** We now present our rounding algorithms. Given an optimal solution to VP, we round the solution to  $C_G$  vectors to find a feasible channel assignment. We have to make sure that the constraints on the number of colors used by  $v$  are satisfied. To round the solution, we select  $C_G$  random vectors, denoted as  $R = \{r_1, r_2, \dots, r_{C_G}\}$ . Each random vector  $r_i = (r_{i,1}, r_{i,2}, \dots, r_{i,m})$  is selected by choosing each component  $r_{i,j}$  independently at random from a standard normal distribution  $N(0, 1)$ .

We consider two rounding algorithms.

- **SDP-COLORSET** (Algorithm 3): We first choose  $C_v$  closest vectors for each vertex  $v$ , and then assign vectors to edges. For each vertex  $v$ , we choose a set  $S_v \subseteq R$  such that  $|S_v| = C_v$  and  $r_i \cdot Y_v \geq r_j \cdot Y_v$  for any  $r_i \in S_v, r_j \notin S_v$ . Given sets  $S_v$ , we assign vectors to edges. For an edge  $e = (u, v)$ , we choose the closest vector  $r_i$  from  $S_u \cap S_v$  (i.e.,  $r_i \cdot X_e \geq r_j \cdot X_e$  for any  $r_j \in S_u \cap S_v$ ). If  $S_u \cap S_v$  is empty, then the edge is dropped.
- **SDP-GREEDY** (Algorithm 4): We greedily choose vectors for each edge from  $R$  instead of choosing color sets for nodes first. Unlike **GREEDY ALGORITHM** where we try to minimize the number of conflicts to be created, we use the solution to SDP to pick a vector for each edge. That is, we find the closest vector  $r_i$  to  $X_e$  in the set  $S$  of colors that can be used for the edge. As in **GREEDY ALGORITHM**,  $S$  is defined to be  $R$  if  $|S_u| < C_u$  and  $|S_v| < C_v$ , and otherwise if either of sets is tight, we choose the set as  $S$ . If both are tight,  $S = S_u \cap S_v$ . The edge will be dropped when  $S$  is empty.

## 4 Experimental Results

### 4.1 Settings

**Network graphs:** We generated network graphs  $G = (V, E)$  as follows. We place nodes in 1000 by 1000 square

---

**Algorithm 4** SDP-GREEDY ALGORITHM

---

```
solve VP and obtain  $\{X_e, Y_v\}$ .
randomly select  $k$  vectors (Let  $R = \{r_1, r_2, \dots, r_k\}$  be the chosen vectors).
for each edge  $e$  do
  if  $|S_v| = C_v$  and  $|S_u| = C_u$  then
     $S = S_u \cap S_v$ .
  else if  $|S_v| = C_v$  and  $|S_u| < C_u$  then
     $S = S_v$ .
  else if  $|S_v| < C_v$  and  $|S_u| = C_u$  then
     $S = S_u$ .
  else
     $S = R$ .
  end if
if  $S = \emptyset$  then
  drop edge  $e$ .
else
  choose the vector with maximum value of  $r_i \cdot X_e$  from  $S$ .
  assign color  $r_i$  to edge  $e$ .
  add  $r_i$  to  $S_v$  and  $S_u$  if not included.
end if
end for
```

---

area uniformly at random, and create an edge between two nodes if they are in transmission range ( $R_t$ ). That is, a pair of nodes has a link between them if the distance is no greater than  $R_t$ . We only use graphs that are connected.

**Number of Channels:** We vary the number of channels available in the network ( $C_G$ ) from 2 to 8. For  $C_v$ , we consider three different types of distributions. In addition, we restrict  $\max C_v \leq 3$  as it is unlikely that a node has more than 3 interface cards in realistic settings.

- *Homogeneous distribution:* Every node  $v$  has the same number of interface cards.  $C_v = k$  for all  $v$ . We represent this distribution as  $h(l)$  where  $l$  is  $\max C_v$ .
- *Uniform distribution:* Node  $v$  can have different number of interface cards. We assign  $C_v$  uniformly at random from  $\{1, \dots, \max C_v\}$  for each node  $v$ . We represent this distribution as  $u(l)$  where  $l$  is  $\max C_v$ .
- *Proportional distribution:* Each node has the number of interface cards proportional to its degree. We consider  $C_v = \lceil d_v/2 \rceil$  and  $\lceil d_v/3 \rceil$  where  $d_v$  is the degree of node  $v$ . That is, the network administrator may assign more wireless cards to nodes with high degrees (hub nodes) to handle the network traffic. We represent this distribution as  $p(l)$  where  $l$  is either  $d_v/2$  or  $d_v/3$ .

**SDP-based Algorithm:** We solve the semidefinite programming using SDPA solver [21]. Given an optimal solution to SDP, we run each rounding algorithm 10 times and choose the best results among the solutions. When we choose the best results, we first look at the number of edges dropped as we want to minimize the edges drops and avoid the network partition, and then choose the solution with minimum conflicts among the solutions with the least number of edge dropped.

## 4.2 Results

We present the experimental results mainly for the graphs generated with 25 nodes and 370 transmission range.

We randomly generated 3 instances for each  $C_v$  distribution and present the average performance of them. The characteristics of each graph are shown in Table 1 .

Instance	# edges	avg degree	max degree	min degree	# possible conflicts
(I)	82	6.56	11	2	1975
(II)	82	6.56	10	3	1730
(III)	78	6.24	9	1	1456

Table 1: Characteristics of network instances. The number of possible conflicts implies the number of pairs  $e_1, e_2$  such that  $I(e_1, e_2) = 1$ .

**Number of conflicts:** We first compare the number of conflicts. When different subset of edges are dropped in channel assignments, we cannot directly compare the number of conflicts since the number of conflicts will decrease as more edges are dropped. Therefore, we first consider the instances where no edge is dropped. Figure 5 compares the ratio of the number of conflicts to the lowerbound that we obtained from SDP (3.2)–(3.6). The figure includes the homogeneous distributions when  $C_G \leq 5$ , and the proportional distribution when  $C_G = 5$  and  $C_v = d_v/2$ . We found that GREEDY ALGORITHM and SDP-BASED ALGORITHMS outperform NAIVE ALGORITHM. Moreover, the algorithms give solutions close to the optimal. Among three algorithms (except NAIVE ALGORITHMS), SDP-GREEDY ALGORITHM maintains more edges than other algorithms for most of instances. Figure 6 compares only NAIVE ALGORITHM and SDP-GREEDY ALGORITHM for the cases when SDP-GREEDY ALGORITHMS keep all edges (GREEDY ALGORITHM and SDP-COLORSET ALGORITHM cannot be compared due to edge drops).

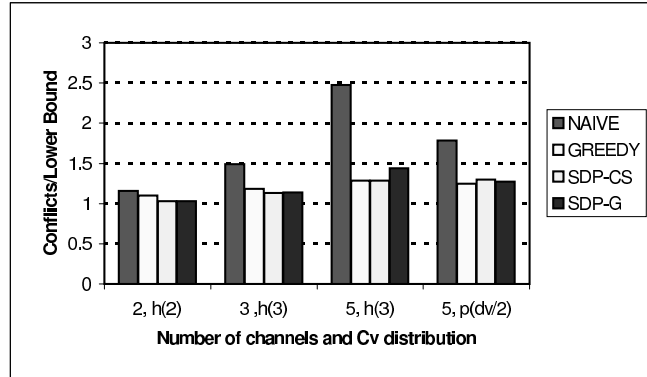


Figure 5: The ratio of the number of conflicts to the lowerbound when no edge is dropped in all methods.  $(a, b)$  in x-axis represents that  $C_G = a$  and the distribution of  $C_v$  follows  $b$ .

Figure 8 - 10 in Appendix show the ratio of the number of conflicts to the lowerbound for all distributions of  $C_v$ . The number of edges dropped is presented on top of the corresponding bar. In most of cases, GREEDY ALGORITHM gives the lowest number of conflicts but it is mainly because the algorithm drops many edges. Note that when the number of conflicts is reduced by dropping edges, it does not necessarily mean that the performance is improved since if an edge is dropped, then the pair of nodes will have to use another route to communicate with each other, thus increasing the traffic. The more accurate performance when edges are dropped would require packet-level simulations (e.g., using *ns-2*).

Figure 7 shows the results with a wing graph as shown in Figure 3. As expected, the performance of SDP-based Algorithms dramatically improves as the number of interface cards in the hub node (black node) increases while the results for NAIVE ALGORITHM remains the same. The greedy algorithm shows the smaller number of conflicts but it drops 33 % (8 out of 24) of edges.

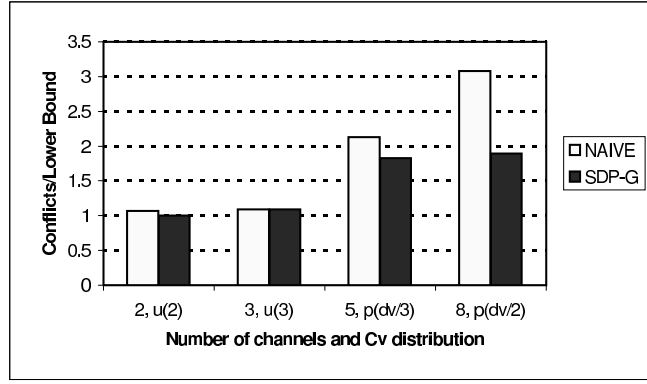


Figure 6: The ratio of the number of conflicts to the lowerbound: comparison of NAIVE ALGORITHMS and SDP-GREEDY ALGORITHM.  $(a, b)$  in x-axis represents that  $C_G = a$  and the distribution of  $C_v$  follows  $b$ .

**Connectivity:** We compare the number of edges dropped in Table 2. We include ‘N’ in parenthesis if the network is disconnected. For all algorithms except NAIVE ALGORITHM, the number of edge dropped increases as  $C_G$  increases or when the network is heterogenous. The greedy algorithm drops more edges than other algorithms in general and sometimes the network is disconnected since in the algorithm each edge chooses its channel only with local information. Among two SDP-based algorithms, more edges are dropped in SDP-COLORSET ALGORITHM. This is because in SDP-COLORSET ALGORITHM, each node independently chooses its color set without considering the information in the neighborhood (even though it is guided by the solution to SDP). Therefore, as  $C_G$  increases, the chances that the sets for two endpoints of an edge are disjoint increase.

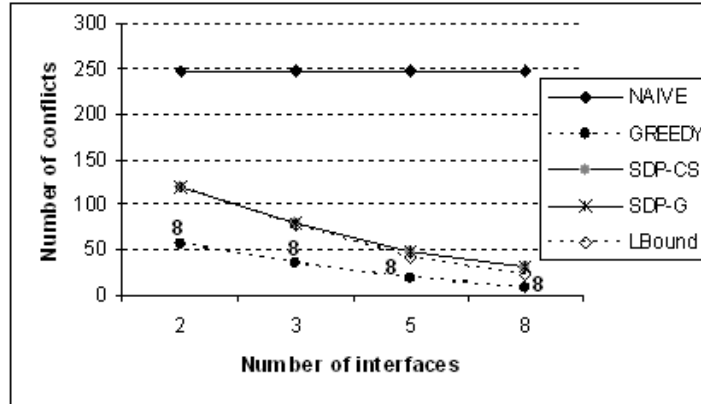


Figure 7: The number of conflicts for the example shown in Figure 3. The number beside each point indicates the number of edges dropped.

In GREEDY ALGORITHM, we had several cases where the network is disconnected especially when  $C_G \geq 5$  and the distribution of  $C_v$  is heterogenous. For SDP-based Algorithm, we had only one instance disconnected with SDP-COLORSET ALGORITHM while all solutions are connected in SDP-GREEDY ALGORITHM. In multi-hop wireless networks, it may be acceptable to drop a subset of edges as communications can occur through other

Instance: Method:	(I)			(II)			(III)			
	greedy	SDP-CS	SDP-G	greedy	SDP-CS	SDP-G	greedy	SDP-CS	SDP-G	
Homo	(2,2)	0	0	0	0	0	0	0	0	
	(3,3)	0	0	0	0	0	0	0	0	
	(5,3)	0	0	0	0	0	0	0	0	
	(8,3)	8	10	0	10	15	2	11	14	2
Uniform	(2,2)	11	0	0	9	0	0	11	0	0
	(3,3)	17(N)	0	0	8	0	0	12	0	0
	(5,3)	15	6	3	6	2	1	12(N)	3	2
	(8,3)	12	1	0	18	3	1	17	11	1
Prop	(5, $d_v/2$ )	1	0	0	0	0	0	0	1	0
	(5, $d_v/3$ )	6	3	1	5	12	4	8(N)	6	0
	(8, $d_v/2$ )	1	7	0	3	7	1	7	9	0
	(8, $d_v/3$ )	9	13	8	5	18	7	8(N)	17(N)	6

Table 2: The number of edge dropped. We include ‘N’ in parenthesis if the network is disconnected.  $(a, b)$  in each row represents that  $C_G = a$  and the distribution of  $C_v$  follows  $b$ .

paths. However, dropping too many edges is not desirable as it may increase the traffic in other links, and more importantly, the network should remain connected after dropping edges.

**Time and space requirements:** NAIVE and GREEDY ALGORITHM require  $O(|E|^2)$  time and  $O(|V| \cdot C_G + |E|)$  memory space. SDP-BASED ALGORITHMS require  $O(C_G \cdot (|V| + |E|))$  time except the time for solving SDP and memory space proportional to the number of possible interference pairs (i.e., pairs of  $e_1, e_2$  such that  $I(e_1, e_2) = 1$ ). The running time of SDP solver depends on the number of possible interference pairs and the precision. In the desktop PC (with 2 Gigabyte memory and Intel Pentium IV 3.6GHz CPU), SDP-based algorithms require the average of 20 minutes for the instances to obtain the optimal solution to SDP while all other algorithms can be finished in less than one second. The SDP solver used more than 1.5 Gigabyte memory when the number of possible interference pairs (i.e., pairs of  $e_1, e_2$  such that  $I(e_1, e_2) = 1$ ) is around 2000.

## References

- [1] M. Mahdian, “the computational complexity of strong edge coloring,” *Disc. Appl. Math.*, vol. 118, pp. 239–248, 2002.
- [2] Christopher L. Barrett, Gabriel Istrate, V.S. Anil Kumar, Madhav V. Marathe, Shripad Thite, and Sunil Thulasidasan, “Strong edge coloring for channel assignment in wireless radio networks,” in *4th IEEE Conference on Pervasive Computing and Communications Workshops (PERCOMW’06), (Workshop on Foundations and Algorithms for Wireless Networking, Pisa, Italy)*, 2006, pp. 106–110.
- [3] R. Faudree, A. Gyarfás, R. Shelp, and Z. Tuza, “The strong chromatic index of graphs,” *Combinatoria*, vol. 29B, pp. 205–211, 1990.
- [4] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, “A multi-radio unification protocol for IEEE 802.11 wireless networks,” Tech. Rep., Microsoft Technical Report, MSR-TR-2003-41, June 2003.
- [5] Min ho Shin, Seungjoon Lee, and Yoo-Ah Kim, “Distributed channel assignment for multi-radio wireless networks,” in *The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006.
- [6] Ashish Raniwala, Kartik Gopalan, and Tzi-cker Chiueh, “Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks,” *SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 50–65, 2004.
- [7] A. Raniwala and T. Chiueh, “Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network,” in *Proceedings of Infocom*, March 2005.
- [8] Mansoor Alicherry, Randeep Bhatia, and Li (Erran) Li, “Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks,” in *In Proceedings of ACM MobiCom*, 2005.
- [9] Yoo-Ah Kim, Seungjoon Lee, and Min ho Shin, “Soft edge coloring,” Manuscript.
- [10] Sven Oliver Krumke, Madhav V. Marathe, and S. S. Ravi, “Models and approximation algorithms for channel assignment in radio networks,” *Wireless Networks*, vol. 7, no. 6, pp. 575–584, 2001.

- [11] V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan, “End-to-end packet-scheduling in wireless ad-hoc networks,” in *Proceedings of SODA*. 2004, pp. 1021–1030, SIAM.
- [12] P. Gupta and P. Kumar, “Capacity of wireless networks,” *IEEE Trans. on Information Theory*, pp. 388–404, March 2000.
- [13] A. FRIEZE and M. JERRUM, “Improved approximation algorithms for MAX k-CUT and MAX BISECTION,” in *Integer Programming and Combinatorial Optimization*, Egon Balas and Jens Clausen, Eds., vol. 920, pp. 1–13. Springer, 1995.
- [14] D. Karger, R. Motwani, and M. Sudan, “Approximate graph coloring by semidefinite programming,” *Journal of the ACM*, vol. 45, pp. 264–265, 1998.
- [15] F. Alizadeh, “Interior point methods in semidefinite programming with applications to combinatorial optimization,” *SIAM Journal on Optimization*, vol. 5, no. 1, pp. 13–51, 1995.
- [16] M. Grotschel, L. Lovasz, and A. Schrijver, “The ellipsoid method and its consequences in combinatorial optimization,” *Combinatorica*, vol. 1, pp. 169–197, 1981.
- [17] M. Grotschel, L. Lovasz, and A. Schrijver, “Geometric algorithms and combinatorial optimization,” *Springer-Verlag*, 1987.
- [18] V. Nesterov and A. Nemirovskii, “Self-concordant functions and polynomial time methods in convex programming,” *Central Economical and Mathematical Institute, U.S.S.R. Academy of Science, Moscow*, 1990.
- [19] V. Nesterov and A. Nemirovskii, “Interior-point polynomial algorithms in convex programming,” *Society for Industrial and Applied Mathematics(SIAM)*, 1994.
- [20] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.
- [21] “Sdp online solver,” <http://grid.r.dendai.ac.jp/sdpa/index.html>.

## A Appendix

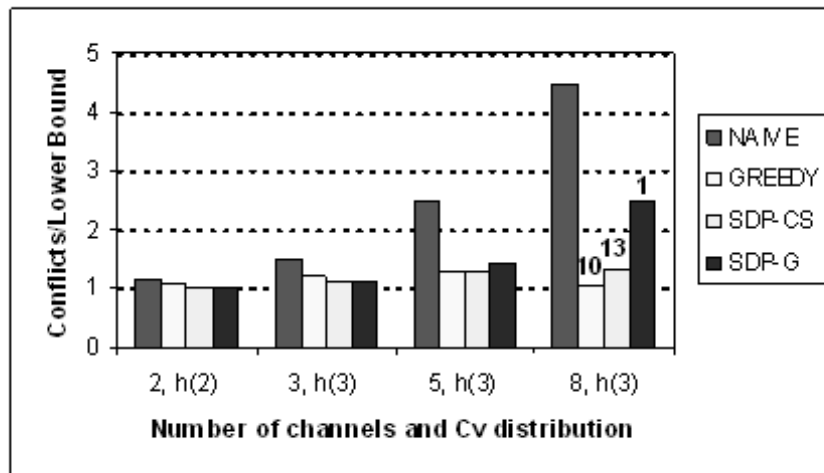


Figure 8: The average ratio of the number of conflicts to the lowerbound for homogeneous distribution. All nodes have the same number of interface cards.  $(a, b)$  in x-axis represents that  $C_G = a$  and the distribution of  $C_v$  follows  $b$ .

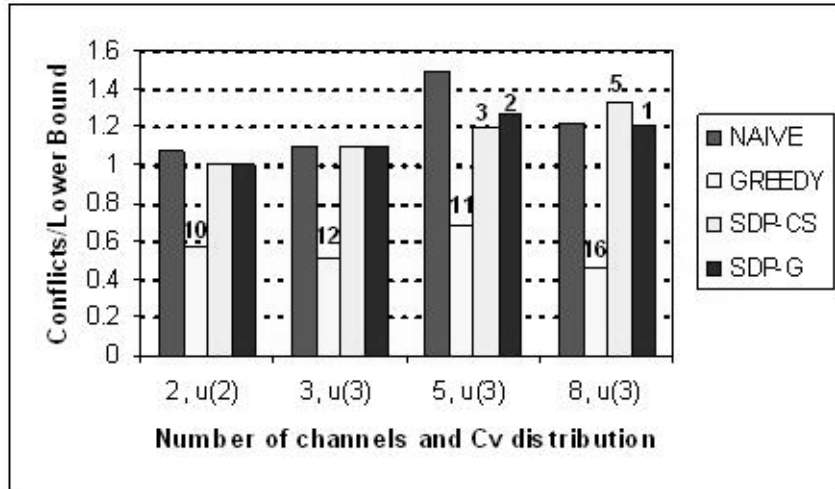


Figure 9: The average ratio of the number of conflicts to the lower bound for uniform distribution. We assign  $C_v$  to each node uniformly at random from  $\{1 \dots 3\}$ .  $(a, b)$  in x-axis represents that  $C_G = a$  and the distribution of  $C_v$  follows  $b$ .

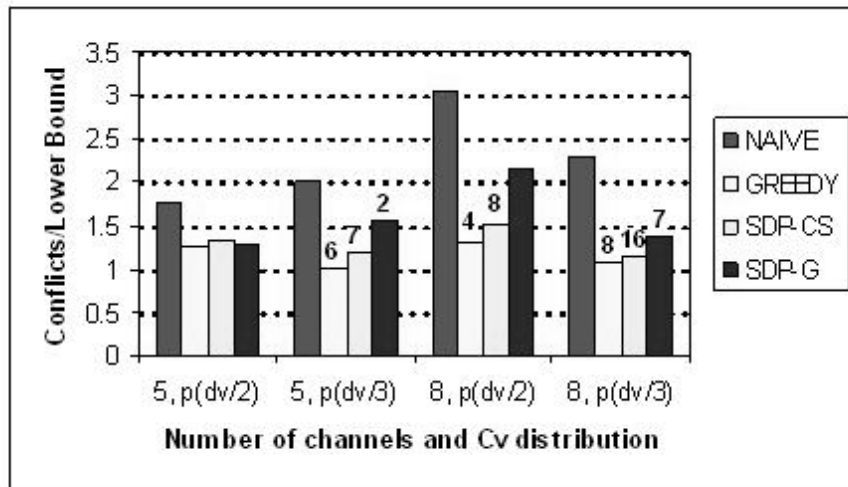


Figure 10: The average ratio of the number of conflicts to the lower bound for proportional distribution. Each node has  $C_v$  proportional to its degree. We have  $C_v = d_v/2$  or  $C_v = d_v/3$ .  $(a, b)$  in x-axis represents that  $C_G = a$  and the distribution of  $C_v$  follows  $b$ .