Multi-Channel Assignment and Link Scheduling for Prioritized Latency-Sensitive Applications

Shih-Yu Tsai[1], Hao-Tsung Yang[1], Kin Sum Liu[1], Shan Lin[2], Rezaul Chowdhury[1], and Jie Gao[1]

[1] Department of Computer Science, Stony Brook University, USA
{shitsai,haotyang,kiliu,rezaul,jgao}@cs.stonybrook.edu
[2] Department of Electrical and Computer Engineering, Stony Brook University, USA
shan.x.lin@stonybrook.edu

Abstract. Current wireless networks mainly focus on delay-tolerant applications while demands for latency-sensitive applications are rising with VR/AR technologies and machine-to-machine IoT applications. In this paper we consider multi-channel, multi-radio scheduling at the MAC layer to optimize for the performance of prioritized, delay-sensitive demands. Our objective is to design an interference-free schedule that minimizes the maximum weighted refresh time among all edges, where the refresh time of an edge is the maximum number of time slots between two successive slots of that edge and the weights reflect given priorities. In the single-antenna unweighted case with k channels and n transceivers, the scheduling problem reduces to the classical edge coloring problem when $k \geq \lfloor n/2 \rfloor$ and to strong edge coloring when $k = 1$, but it is neither edge coloring nor strong edge coloring for general k. Further, the priority requirement introduces extra challenges. In this paper we provide a randomized algorithm with an approximation factor of $\tilde{O}\left(\max\left\{\sqrt{\Delta_p}, \frac{\Delta_p}{\sqrt{k}}\right\}\log m\right)$ in expectation, where Δ_p denotes the maximum degree of the unweighted multi-graph, which is formed by duplicating each edge e_i for w_i times (w_i is e_i's integral priority value), and m is the number of required link communications.[3] The results are generalized to the multi-antenna settings. We evaluate the performance of our methods in different settings using simulations.

Keywords: Latency sensitive scheduling · Multi-channel scheduling · Fairness.

1 Introduction

Today's communication networks have provided great support to delay-tolerant applications (e.g., web, email). But demands for latency-sensitive applications in wireless and mobile networks are rising, with emerging applications from video-conferencing, real-time interactions using Virtual Reality/Augmented Reality (VR/AR), vehicular networking and distributed robotics. These new applications require more stringent delay guarantees. To support latency-sensitive applications, one must develop network control algorithms at various layers with latency guarantees.

[3] $f(n) \in \tilde{O}(h(n))$ means that $f(n) \in O\left(h(n)\log^k(h(n))\right)$ for some positive constant k.

In this paper, we look at the MAC layer and consider a TDMA-based (Time-Division Multiple Access) multi-channel link scheduling problem. Multi-radio multi-channel architecture is widely adopted in wireless mesh networks deployments (e.g., in MIT Roofnet, WING [1, 4, 5, 8, 24]) and is increasingly supported in IEEE standards (e.g., 02.11 and 802.16) [7, 9, 11]. We assume that there are k channels of different frequencies to use and each node may have one or multiple radio interfaces, possibly operating on different channels. At each time slot, each of the radio interfaces may be assigned one of the k channels and if two nodes within the same communication range have two radio interfaces on the same channel, the message can be successfully received provided there is no interference in the neighborhood. The general question on channel assignment and scheduling is to decide for each link which channel to use and when, given an optimization objective. In this paper, we examine the following problem.

Min Max Weighted Refresh Time Scheduling. Given k channels and a simple weighted graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ in which edge $e_i \in E$ has integer weight w_i with the minimum edge weight being 1, we would like to design a periodic schedule for all edges in G,[4] which specifies a set of edges for each time slot and the channels they use. The channel assignment for an edge (u, v) specifies the channel that the transceivers at u and v adopt. If a node has r radio interfaces, different radio interfaces may operate on different channels. A feasible schedule must follow the following rules to avoid interference:

– There are at most r active edges incident to any node at any given time, since a node has r radio interfaces.
– Two edges that are active at the same time must use different channels if they are within the interference distance from each other, i.e., they have a common endpoint or some of their endpoints are neighbors[5].

Here, we consider interference at the protocol level, leaving the physical model (SINR model) for future exploration. Our goal is to find a feasible schedule of all edges that minimizes

$$\max_{i \in \{1,\ldots,m\}} w_i T_i.$$

Here, T_i denotes the *maximum refresh time* for edge e_i, i.e., the maximum number of time slots until edge e_i appears again in the given schedule. We name this problem as Min Max Weighted Refresh Time Scheduling problem for the case of non-uniform weights; Min Max Refresh Time Scheduling otherwise.

In this paper, we focus on the algorithmic aspect of the scheduling problem and assume that the networking issues (synchronization, packet loss, and re-transmissions) are handled in the standard manner. We assume relatively long streaming traffic flow such that the schedules for traffic demands are updated when traffic demands change substantially.

[4] Schedules are restricted to be periodic because each non-periodic infinite schedule with a finite max weighted refresh time can be turned to a periodic schedule with the same refresh time. See Appendix for a proof.

[5] This is the case of ℓ-hop interference model (wireless links $\ell + 1$ or more hops away from one another can be scheduled to transmit data at the same time) when $\ell = 2$.

Motivation and Related Work. A lot of prior work on channel assignment and scheduling focused on maximizing network throughput (i.e., the total number of links one can schedule in a single slot without interference), or makespan (i.e., minimize the time slots to complete a given demand vector), which will be reviewed in the next section. For latency sensitive applications fairness is important as well, in order for traffic flows to experience steady and predictable latency over time. Our problem provides guaranteed share of resources for each edge. Further, we wish to allow prioritized treatment for emergency oriented applications (compared to recreational applications). This can be implemented by edges on the routes of traffic with high priority carrying higher weights. In our problem, these edges are scheduled more frequently.

Mathematically, our problem is closely related to edge coloring and strong edge coloring problems. The problem of edge coloring is to assign a color to each edge such that no two adjacent edges have the same color. The minimum number of colors used is called the *chromatic index*, which is either $\varDelta$ or $\varDelta + 1$, where $\varDelta$ is the maximum degree in the graph (Vizing's Theorem), although deciding which one is the optimal index is NP-hard [13]. Greedy coloring, i.e., use a color that is not yet used in the neighboring edges, gives a 2-approximation. In strong edge coloring, two edges e, e' cannot have the same color if they share a common endpoint or their endpoints are connected by an edge (Figure 1). In the wireless network setting, this maps to the scenario when protocol level interference is considered and shall be eliminated in the schedule [3, 17, 20, 21]. The minimum number of colors used is called the *strong chromatic index*. Counting the number of edges that could be in conflict with any edge shows that the strong chromatic index is between $\varDelta$ and $2\varDelta(\varDelta - 1) + 1$. Erdös and Nešetřil conjectured that the strong chromatic index is at most $5\varDelta^2/4$, which is still open. For a given graph, computing its strong chromatic index is NP-hard [25], and a greedy algorithm gives a $\Theta(\varDelta)$ approximation. Closing the gap appears to be a long-standing problem (see [16]).

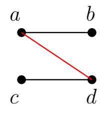


Fig. 1. The edges (a, b), (c, d) can be colored the same in edge coloring but cannot be colored the same in strong edge coloring (due to the edge (a, d)).

Our problem adds more complications by considering k possible channels and r radios per node. As we will show in this paper, our scheduling problem includes edge coloring and strong edge coloring as special cases. As far as we are aware, our problem has not been studied before.

Our contribution. In this paper, we initiate the study of the Min Max Weighted Refresh Time Scheduling problem. We consider the single radio case first, i.e., $r = 1$. When edges have the same weight, our problem reduces to the classical *edge coloring* problem if k is at least $\lfloor n/2 \rfloor$ and the *strong edge coloring* problem if $k = 1$. For a general k, a greedy algorithm that assigns each edge the earliest possible time slot with the first available channel achieves an approximation factor of $\left\lceil \frac{2(\varDelta-1)}{k} \right\rceil + 2$, where $\varDelta$ is the maximum degree of the given graph. Notice that this bound is a smooth transition from the 2-approximation for edge coloring to $\Theta(\varDelta)$-approximation for strong edge coloring, when k varies between 1 and $n/2$.

When edges have different priorities/weights, the problem becomes tricky. Intuitively, an edge with a higher weight should be scheduled more frequently. That is, we may want to create multiple copies of this edge so that we can apply the scheduling algorithm for the unweighted setting by treating each duplicate edge as a different edge. But how many copies should we make for an edge of weight w_i? Second, the duplicated copies of the same edge, ideally, shall be spread uniformly in the schedule, avoiding a large gap somewhere. However, it is not clear how to ensure the uniform placement of the duplicated copies of e_i, for every edge e_i, with non-trivial interference patterns to avoid. Last, we need to obtain a lower bound for the optimal refresh time in order to prove approximation factors.

Our insights come from understanding the optimal schedule. Suppose the optimal schedule repeats every T slots. There is a lower bound $\mathbb{L}(S)$ of the optimum maximum refresh time – by simply dividing T by μ_i, the number of times e_i appears in one cycle, for each edge e_i. Next, we show that this lower bound achieves the minimum value ℓ^* if μ_i is Cw_i, for some integer C. This is useful for the algorithm design as we know that the number of copies duplicated for e_i shall be *proportional* to w_i, but we still do not know what C is. By using the probabilistic method, we show that if we set $C = 1$ and take the schedule that minimizes the lower bound $\mathbb{L}(S)$, then the value of $\mathbb{L}(S)$ is at most a factor of $7 \log m$ of ℓ^*, where m is the number of distinct edges to be scheduled. This way we are only losing a factor of $O(\log m)$.

The analysis above suggests the following simple scheduling algorithm for the weighted setting. We first make w_i copies of edge e_i, generate a random permutation of these edges (possibly with duplicates), and partition the permutation into chunks of equal length. For each chunk, run the aforementioned greedy scheduling algorithm and then combine the schedules together. We ensure that the length of each chunk is small enough such that for each edge e, only $O(1)$ edges in expectation may interfere with e. Hence, the greedy schedule uses $O(1)$ time slots for each chunk. Further, in a random permutation, the duplicated edges are likely to be placed evenly – the maximum gap can be bounded by the standard balls and bins problem. In summary, the approximation factor (in expectation) is bounded as

$$O\left(\max\left\{\sqrt{\varDelta_p}, \frac{\varDelta_p}{\sqrt{k}}\right\} \log m \frac{\log W_{\max}}{\log\log W_{\max}}\right),$$

where $\varDelta_p$ denotes the maximum degree of the unweighted multi-graph, which is formed by duplicating each edge e_i for w_i times, and $W_{\max}$ is the highest weight of all edges. Notice that the endpoints of the edge of maximum weight has degree at least $W_{\max}$ in the multi-graph. That is, $\varDelta_p \geq W_{\max}$. Hence, the provided approximation factor can be written concisely as $\tilde{O}\left(\max\left\{\sqrt{\varDelta_p}, \frac{\varDelta_p}{\sqrt{k}}\right\} \log m\right)$.

Finally, both the weighted and unweighted algorithms can be extended to the multi-antenna case, i.e., $r > 1$. We also run simulations empirically (in Appendix) to evaluate the performance of our algorithms. The simulations show that our unweighted algorithm works as efficiently for large graphs as for small graphs when they have similar densities. When we have a reasonable number of channels, our algorithm can efficiently use them for large graphs to keep the latency low. On the other hand, our weighted algorithm can efficiently use only two available channels for graphs with uniform weight distribution. It is approximately two times better than with only one channel.

The rest of this paper is organized as follows. Section 2 discusses related work. We address the single antenna case in Section 3 and Section 4, and extend our result to the multi-antenna case in Section 5. Section 6 concludes this paper.

2 Related Work

Channel assignment and link scheduling with wireless interference have mainly focused on throughput optimization (maximizing the number of edges that can be scheduled at the same time). This problem is closely related to finding the maximum independent set. For a given demand vector, a commonly formulated problem is to minimize the number of slots to meet the demand, called the *makespan*.

For the centralized setting, Hajek and Sasaki [12] considered the problem of minimizing makespan but ignored wireless interference, proposing two polynomial-time algorithms for direct messages and relayed messages. Ramanathan and Lloyd [19] considered wireless interference and focused on trees and planar graphs. Balakrishnan et al. [2] looked at unit disk graphs and proposed PTAS and distributed constant factor polynomial-time approximation algorithms. Sharma [22] considered approximation algorithms for the k-hop interference model.

A few papers [6, 15] considered fully distributed scheduling algorithms that optimize for throughput or makespan. For 1-hop interference model, the maximum number of edges that could be scheduled at the same time is the maximum matching. A greedy maximal matching algorithm has at least half of edges of the optimal, and in general has an approximation factor depending on the 'interference degree' [6, 14, 28].

The results have been generalized to multi-hop communication scenarios. Kumar et al. [15] studied the problem of minimizing makespan for given packets in a wireless setting with 2-hop interference (the same as ours) and proposed a distributed algorithm with an approximation bound of $\Theta(\Delta \log^2 n)$ for arbitrary graphs. They also show that it is hard to approximate the minimum makespan within a factor of $\Delta^{1-\varepsilon}$ for any positive constant $\varepsilon < 1$, even in the centralized setting. On the other hand, with the same greedy idea, Wan et al. [26] scheduled replicated edges (traffic demands on direct-communicated links) in any multi-hop wireless network under any arbitrary interference model. The proposed algorithm achieves a $1 + \mu \ln \alpha$ approximation ratio using a $\mu-$approximate algorithm for finding a maximal set of transmitting edges to greedily schedule the edges, where α is the maximum number of edges that can transmit simultaneously. Furthermore, in the multi-antenna scenario under the binary interference model, they also considered a variant in which traffic demands are given on the node-level links and proposed a constant factor approximation algorithm [27].

Fairness is not considered in the scheduling literature as much as throughput. Shi et al. [23] discussed the existing fairness models of channel assignment and compared them systematically. They also stated several challenges, such as designing fairness strategies under distributed scenarios (since we consider wireless networks), corrective strategies for unfairness, and how to assign weights to nodes and how to allocate resources according to the weights. Most studies have focused on resource allocation but the weight assignment strategies have not received much attention. Chaporkar et al. [6] proposed the use of a token generation mechanism together with maximal scheduling for fairness, but no guarantee is provided.

3 Min Max Refresh Time in the Single-Antenna Setting

We start with the case in which all edges are unweighted and each node has only one antenna ($r = 1$) and show the connection of our problem (i.e., the Min Max Refresh Time Scheduling problem defined in Section 1) with other graph problems. For different k, the number of channels, the problem in the single-antenna unweighted case maps equivalently to different graph coloring problems.

– When $k = 1$, this problem is equivalent to the *strong edge coloring* problem. The edges of the same color are scheduled during the same time slot.
– When $k \geq \lfloor |V|/2 \rfloor$, the problem is equivalent to the *edge coloring* problem, where V is the set of transceivers. Again the edges of the same color are scheduled during the same time slot – though they may use different channels.
– In between the problem is neither edge coloring nor strong edge coloring. We show that a greedy algorithm gives a $\lceil 2(\Delta - 1)/k \rceil + 2$ factor approximation to the optimal solution.

Theorem 1. *In the single-antenna case with unit weights, the Min Max Refresh Time Scheduling problem with only one channel available is equivalent to strong edge coloring.*

Proof. In a strong edge coloring problem, the edges of the same color form an induced matching. Let us identify the colors by unique integers from $\{1, \cdots, c\}$. Here c is the number of available colors. We schedule all edges of color i during time slot i, for all $i \in \{1, \cdots, c\}$. We repeat this finite schedule forever to form our final infinite schedule. In each slot, the edges do not cause any interference. Further, the maximum refresh time for any one edge is exactly c in this schedule.

In the other direction, given an infinite schedule solution with the maximum refresh time t, it can be transformed into a periodic schedule by finding the prefix schedule that achieves the maximum refresh time. Focus on this prefix schedule (cycle), it will create an induced matching for each slot and we remove any duplicate edges in this cycle. If we color the edges in the same time slot by the same color, this becomes a valid strong edge coloring solution. The maximum refresh time t implies that the cycle cannot have a length more than t, so the valid strong edge coloring solution has at most t colors. Therefore, the two problems are equivalent. $\square$

Theorem 2. *In the single-antenna case with unit weights, the Min Max Refresh Time Scheduling problem with at least $\lfloor |V|/2 \rfloor$ channels is equivalent to the edge coloring problem.*

Proof. Suppose we are given an edge coloring solution with c colors. Observe that it is a decomposition of the given graph G into c matchings. Assign a new time slot to each matching. In each time slot, assign each edge in the corresponding matching to a different channel. There are at most $\lfloor |V|/2 \rfloor$ edges in a matching of G so we have enough channels to build this schedule. Form a periodic schedule by repeating this schedule of c time slots. The refresh time for any edge is at most c.

On the other hand, given an infinite schedule with maximum refresh time t, it must have a smallest periodic cycle of length at most t. By eliminating any duplicate edges, each edge appears exactly once now, i.e., this cycle partitions the edges into at most t time slots. The edges scheduled for a given time slot cannot share any common vertices – since each node is given only one channel. That means each time slot gives a matching, so this graph is t edge colorable. □

For any k, we show that the following greedy algorithm has an approximation ratio of $\lceil 2(\Delta - 1)/k \rceil + 2$. We examine the edges one by one. For each edge e, we check the first slot with the first channel to see if e can be scheduled without violating any constraints. If not, we move on to the next channel and check again. If we run out of channels, we move on to the next time slot. When we go through all the edges, denote the number of slots used as h. We then repeat the schedule an infinite number of times. The refresh time for all edges is precisely h.

To show the approximation factor, we observe that the optimal schedule for G, for any k, is at least Δ – this is because these Δ edges attached to the common node must be placed in different slots.

Theorem 3. *In the single-antenna unweighted case, the greedy algorithm gives a schedule with a maximum refresh time of at most* $\lceil \frac{2(\Delta-1)^2}{k} \rceil + 2(\Delta - 1) + 1$. *Therefore, this algorithm is a* $\lceil \frac{2(\Delta-1)}{k} \rceil + 2-$*approximate algorithm for the Min Max Refresh Time Scheduling problem.*

Proof. We consider the edges that are placed in the last time slot of the generated finite schedule. Take one of these edges, say, edge e. The reason e is placed at the h-th slot, by the greedy rule, is that it cannot be placed anywhere earlier. For each of the previous slot, at least one of the following two events happens: the first event is that an edge incident to one endpoint of e is scheduled and so we cannot schedule e. The second event is that we run out of channels for e. For *each* of the k channels, there is an edge e' that is at most one hop away from e and is scheduled with this channel. (Here, at most one hop away means that edges e and e' have a common endpoint or some of their endpoints are neighbors.)

Since e is incident to at most $2(\Delta - 1)$ edges, the number of slots of the first type is at most $2(\Delta - 1)$. At most $2(\Delta - 1)^2$ edges are one hop away from e. For the second event to happen we need to use k such edges. Therefore, there are at most $\lceil 2(\Delta - 1)^2/k \rceil$ slots of the second type.

The worst case happens if, for each slot, exactly one of the two events happens. This amounts to a total of at most $\lceil 2(\Delta - 1)^2/k \rceil + 2(\Delta - 1) + 1$ slots. Furthermore, the lower bound of optimal max refresh time Δ means that this greedy algorithm produces a ($\lceil 2(\Delta - 1)/k \rceil + 2$)-approximate solution. □

Remark. When $k = \Theta(\Delta)$, the greedy algorithm is a constant approximation. The upper bound $\lceil 2(\Delta - 1)^2/k \rceil + 2(\Delta - 1) + 1$ is nearly tight – for $k = 1$ there are graphs that require $\Omega(\Delta^2)$ slots. For example, a 5-cycle must use 5 colors for strong edge coloring. If we glue two 5-cycles together as in Figure 2, the graph requires 20 colors. In general, if we glue l of the 5-cycles together, this amounts to $5\Delta^2/4$, where $\Delta = 2l$.

Remark. The bounds here work for general graphs. Specifically, in the case of unit disk graphs, using packing argument, we can show that the greedy algorithm mentioned above has a $O(1)$ approximation factor [3].

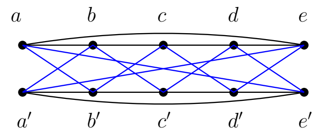


Fig. 2. Two 5-cycles $abcde$ and $a'b'c'd'e'$ glued to each other (by the blue edges). The graph requires 20 colors/slots to schedule if a single channel is used.

4 Min Max Weighted Refresh Time in the Single-Antenna Setting

In this section, we discuss the Min Max Weighted Refresh Time Scheduling problem under the single-antenna setting. Let $G = (V, E)$ be a weighted graph, where E is the set of m edges $e_1, ..., e_m$ with weights $w_1, ..., w_m$, respectively. We would like to minimize $\max_i w_i T_i$, where T_i is the maximum refresh time for edge e_i. When w_i's are not the same, the algorithm in the previous section does not work. If an edge is more important, then we would like to schedule it more frequently.

4.1 Lower Bound of the Optimal Solution

We first try to understand the structure of the optimal solution. Let us consider the optimal periodic schedule and consider one cycle S^* of the optimal periodic schedule. Suppose S^* has T^* time slots and the maximum refresh time for edge e_i is T_i^* and edge e_i appears μ_i^* times in S^*. We can picture this periodic schedule as wrapping S^* around on a cycle. T_i^* is the maximum gap between adjacent appearances of i on the cycle. Clearly, $T_i^* \geq T^*/\mu_i^*$ by the pigeonhole principle. Define $\mathbb{L}(S^*) = \max_{i\in\{1,...,m\}} w_i T^*/\mu_i^*$. Therefore, the optimal solution is lower bounded as follows.

$$\mathbb{O}(S^*) = \max_{i\in\{1,...,m\}} w_i T_i^* \geq \max_{i\in\{1,...,m\}} w_i T^*/\mu_i^* = \mathbb{L}(S^*). \tag{1}$$

Now let us suppose we have a collection of edges in which edge e_i is duplicated μ_i times and consider a feasible finite schedule S for these edges and denote by $T(S)$ the total number of time slots. For each feasible finite schedule S, let us define

$$\mathbb{L}(S) = \max_{i\in\{1,...,m\}} w_i T(S)/\mu_i.$$

Here, μ_i is the number of occurrences of edge e_i in S. Now, we want to understand when we can get the minimum value of $\mathbb{L}(S)$ among all feasible finite schedules S. Since S^* is one finite feasible schedule, the minimum value of $\mathbb{L}(S)$ is a lower bound of $\mathbb{L}(S^*)$, hence, a lower bound of $\mathbb{O}(S^*)$ as well.

Lemma 1. *Among all possible feasible schedules S, $\mathbb{L}(S)$ is minimized when the schedule has $\mu_i = Cw_i$ for some integer C, for all i.*

Proof. Assume otherwise. Let S be a schedule that achieves the minimum $\mathbb{L}(S)$ but not every μ_i is exactly Cw_i for some integer C. We will create a feasible schedule S' with $\mathbb{L}(S') < \mathbb{L}(S)$, yielding a contradiction.

First, we repeat the schedule D times for some big integer D, which will be determined later. Now we take the edge e_i that has the largest $w_i T(S)/\mu_i$ among all edges (i.e., i realizes the value $\mathbb{L}(S)$). We create a slot that only contains edge e_i and add this slot at the end of the enlarged schedule. In the new schedule S', we have a total of $T(S') = DT(S) + 1$ slots.

Now we calculate the ratio $w_j T(S')/\mu'_j$, for each edge e_j in the new schedule S'. First, for edge e_i, we have

$$\frac{w_i T(S')}{\mu'_i} = \frac{w_i(DT(S)+1)}{D\mu_i + 1} < \frac{w_i T(S)}{\mu_i} = \mathbb{L}(S)$$

The inequality is true because $\mu_i < T(S)$.

Now consider an edge e_j, $j \neq i$. There are two cases.

– If $w_j T(S)/\mu_j < w_i T(S)/\mu_i$, then we can show that

$$w_j T(S')/\mu'_j < w_i T(S')/\mu'_i = \mathbb{L}(S)$$

by taking

$$D > \frac{w_j}{w_i \mu_j - w_j \mu_i}.$$

– If $w_j T(S)/\mu_j = w_i T(S)/\mu_i = \mathbb{L}(S)$, we repeat the same procedure as above. Notice that in every iteration we remove one edge that realizes $\mathbb{L}(S)$.

At the end we can argue that we find a new schedule S' such that for all edges j, $w_j T(S')/\mu'_j < \mathbb{L}(S)$. Thus $\mathbb{L}(S') < \mathbb{L}(S)$. This is a contradiction to the optimality of S. Hence, the statement of this lemma is true. □

Therefore, $\mathbb{L}(S)$ is minimized when $\mu_i = Cw_i$, for all i and some constant C. Next, we show that it does not hurt too much to consider $\mu_i = w_i$ if we only care about minimizing $\mathbb{L}(S)$.

Suppose we have two scheduling problems, in the first one, each edge e_i is duplicated $\mu'_i = w_i$ times and we take S' to be one of the best finite feasible schedules for these edges that minimize $\mathbb{L}(S')$; while in the second one, each edge e_i is duplicated $\mu''_i = Cw_i$ times, for a variable C taking all possible integer values. Let S'' be one of the optimal schedules that minimize $\mathbb{L}(S'')$. Clearly $\mathbb{L}(S'') \leq \mathbb{L}(S')$ by definition. We now argue that,

Lemma 2. $\mathbb{L}(S') \leq 7 \log m \, \mathbb{L}(S'')$, *where m is the number of edges of G.*

To prove this lemma, we first need the following lemma.

Lemma 3. *Given a bipartite graph with vertex sets X and Y. If all the degree of $y_i \in Y$ is a multiple of C for a fixed constant C and $|X|$ is also a multiple of C, then there is a subset $X' \subseteq X$ such that $|X'| = |X| \cdot \beta/C$ and for each vertex $y \in Y$, the number of neighbors of y in X' is at least $d(y)/C$, where $d(y)$ is the degree of y and $\beta = 7 \log m$ with $m = |Y|$.*

Proof. We will use a probabilistic argument to prove that such an X' exists. Partition the vertex set X into disjoint subsets X_1 and X_2 randomly with $|X_1| = |X|\beta/C$. Hence, each vertex in X has probability β/C to be in X_1. Now, for each vertex $y_i \in Y$, denote by Y_i the number of edges incident to y_i and a vertex in X_1. Clearly the expectation of Y_i is $E(Y_i) = d(y_i) \cdot \beta/C$. By Chernoff bound,

$$\begin{aligned}\text{Prob}\{Y_i \le d(y_i)/C\} &= \text{Prob}\{Y_i \le E(Y_i)\big(1-(1-1/\beta)\big)\} \\ &\le \exp\{-\beta d(y_i)/C \cdot (1-1/\beta)^2/3\}.\end{aligned}$$

Since $d(y_i)/C \ge 1$ and $(1-1/\beta)^2 > 1/2$ for $\beta = 7\log m$,

$$\text{Prob}\{Y_i \le d(y_i)/C\} < \exp\{-\beta/6\}.$$

Therefore, the probability that all nodes in Y have at least $1/C$ fraction of edges in X_1 can be estimated by the union bound.

$$\begin{aligned}&\text{Prob}\{\text{All nodes in } Y \text{ have at least } 1/C \text{ fraction of edges in } X_1\} \\ &\ge 1 - \textstyle\sum_i \text{Prob}\{Y_i \le d(y_i)/C\} > 1 - m/\exp\{\beta/6\} > 0.\end{aligned}$$

Thus, the probability that all vertices in Y have at least $1/C$ of their edges in X_1 is positive. This implies that such a partition must exist. Therefore, such X' exists. □

Now, we are ready to prove Lemma 2.

Proof of Lemma 2. We will prove this by forming a feasible schedule which has edge e_i with occurrence μ_i from schedule S'' and this feasible schedule consists of at most $T(S'')7\log m/C$ time slots, where m is the number of edges of G. (If $T(S'')$ is not a multiple of C, we supplement with empty slots to make it a multiple of C.) Such a feasible schedule exists by Lemma 3. Here, the bipartite graph with vertex set $X \cup Y$ is the following. X consists of $T(S'')$ vertices and each represents a time slot. On the other hand, Y consists of m vertices and each represents an edge in G. We connect a vertex (time slot) in X with a vertex (an edge in G) e_i in Y if the edge e_i in G is scheduled in that time slot. Note that for each vertex $y \in Y$, if its corresponding edge in G is e_i, then the degree of y is exactly Cw_i. Hence, Lemma 3 shows that there is a subset of the time slots in S'' such that each edge e_i appears at least w_i times in these time slots. This generates a scheduling with at most $T(S'')\beta/C$ slots, where $\beta = 7\log m$.

Recall that S' is the optimal schedule with the smallest $T(S')$. Thus, $T(S') \le T(S'')\beta/C$. On the other hand, $\mu' = w_i$ and $\mu'' = Cw_i$ yield $\mathbb{L}(S') = T(S')$ and $\mathbb{L}(S'') = T(S'')/C$. Combining them, we get $\mathbb{L}(S') \le \beta\mathbb{L}(S'') = 7\log m \cdot \mathbb{L}(S'')$. □

Now for the first scheduling problem (we duplicate edge e_i exactly w_i times and minimize $\mathbb{L}(S')$), by the same idea of the max unweighted refresh time in Section 3, we can get a similar lower bound on $T(S')$. Define an unweighted multigraph $G_m = (V, E_m)$ by duplicating edge e_i for w_i times. Let Δ_p denote the maximum degree of G_m. That is, $T(S') \ge \Delta_p$. Combined with the inequality in Lemma 2, we have $\mathbb{L}(S'') \ge \mathbb{L}(S')/(7\log m) = T(S')/(7\log m) \ge \Delta_p/(7\log m)$. On the other hand, as discussed before, the minimum value of $\mathbb{L}(S)$ among all feasible finite schedules S

serves as a lower bound for $\mathbb{O}(S^*)$ and the minimum value occurs when $\mu_i = Cw_i$ for all i, which is the second scheduling problem. Hence, $\mathbb{O}(S^*) \geq \mathbb{L}(S'')$. Now, we get a lower bound on the optimal solution as in Theorem 4.

Theorem 4. *The optimal solution for the Min Max Weighted Refresh Time Scheduling problem has* $\mathbb{O}(S^*) \geq \frac{\Delta_p}{7 \log m}$.

4.2 Algorithm

Our algorithm for the Min Max Weighted Refresh Time Scheduling problem works as follows. First, generate a random permutation of the edges in E_m. Second, partition these edges into $g = \lceil W/b \rceil$ buckets of equal length b, where $W = \sum_i w_i$ and b will be determined later. The last bucket may have length less than b and that is alright. For each bucket B_i, consider the induced unweighted multigraph, G_i, which consists of the edges that appear in this bucket and the edges whose two endpoints appear in edges of this bucket. Observe that G_i includes all edges in B_i but may contain other edges, too (e.g., an edge e that is not in B_i but have both vertices appearing in B_i, but we only include one copy of e if it is a duplicate edge). Finally, run the greedy algorithm in Section 3 on G_i. (In fact, the greedy algorithm works for unweighted multigraphs.) This generates a schedule S_i for all the edges in G_i. Keep only the edges in B_i in the schedule S_i to form a new schedule S'_i. Our final schedule will run $S'_1, S'_2, ..., S'_g$ in sequence and periodically.

4.3 Analysis of the Approximation Ratio

Now we bound the approximation ratio of the aforementioned algorithm. First, we examine how many time slots in expectation are needed for each bucket. Second, we use the balls and bins technique to analyze the maximum weighted refresh time. Let us consider two cases, $k = 1$ and $k > 1$ separately.

Lemma 4. *Suppose $k = 1$ and $b = W/\Delta_p^2$. Then the number of time slots needed for the edges in a specific bucket B_i is $O(1)$ in expectation.*

Proof. The number of time slots needed for a bucket B_i, by the greedy algorithm, depends on the degree of the induced subgraph G_i (Theorem 3). We first analyze the probability that a specific edge $e_j = (u, v)$ falls inside the induced subgraph G_i. In order for this event to happen, either edge e_j is placed in B_i (with probability at most $b/W_{\sum}$ as there are $\lceil W/b \rceil$ buckets in total), or both endpoints u and v of e_j appear in B_j (as other edges incident to u (v) are placed in B_i). We can bound the probability as follows. Recall that $b = W/\Delta_m^2$. By $\lim_{x\to\infty}(1 - 1/x)^x = 1/e$ and Taylor's Formula, we get

$$\begin{aligned}\text{Prob}\{e_j \in G_i\} &\leq b/W_{\sum} + (1 - (1 - b/W)^{\Delta_p})^2 \\ &\leq 1/\Delta_p^2 + (1 - (1 - 1/\Delta_p^2)^{\Delta_p})^2 \\ &\leq 1/\Delta_p^2 + (1 - e^{-1/\Delta_p})^2 \leq 2/\Delta_p^2.\end{aligned}$$

Let edge e_j be the last edge be added to the last time slot in the schedule for G_i. The number of slots used for G_i depends on the number of edges that can interfere with e_j. Specifically, we have to analyze the following two parameters:

– $\Delta_i(e_j)$: the number of edges in G_i, counting duplication, that share a common vertex with e_j.
– $\Delta'_i(e_j)$: the number of edges in G_i, counting duplication, that are exactly one hop away from e_j. That is, some of their endpoints are neighbors and they don't share any common endpoints.

For bounding $\Delta_i(e_j)$ from above, we consider the edges incident to e_j in G_m, each appearing in G_i with probability at most $2/\Delta_p^2$. The total number of these edges other than e_j is $w_j - 1 + 2(\Delta_p - w_j)$. Therefore,

$$E[\Delta_i(e_j)] \le (w_j - 1 + 2(\Delta_p - w_j)) \cdot 2/\Delta_p^2 < 4/\Delta_p \le 4.$$

Similarly, $E[\Delta'_i(e_j)] \le 2\Delta_p^2 \cdot 2/\Delta_p^2 = 4$.

By the linearity of expectation, $E[\Delta_i(e_j) + \Delta'_i(e_j)] = O(1)$. That is, it introduces $O(1)$ edges, in expectation, in the interference range of the last edge e_j. Hence, the number of slots needed to resolve interference for bucket i, in expectation, is bounded by $O(1)$. □

Now, we can analyze the maximum refresh time for each edge e using balls and bins results.

Theorem 5. *With $k = 1$ and $b = W/\Delta_p^2$, the proposed algorithm in Section 4.2 , for the Min Max Weighted Refresh Time Scheduling problem, has an approximation factor of $O(\Delta_p \log m \log W_{\max}/ \log\log W_{\max})$ in expectation, where $W_{\max}$ is the highest possible weight and Δ_p is the maximum degree in G_m.*

Proof. For each edge e_i, we would like to evaluate the maximum refresh time T_i, i.e., the maximum number of time slots before edge e_i is scheduled again. Recall that edge e_i is duplicated w_i times and the schedule is produced from a random permutation of all (duplicate) edges. We examine each gap between adjacent appearances of edge e_i in the permutation (wrapped as a cycle).

The number of edges in this gap can be upper bounded by the balls and bins analysis. Here these w_i duplicated edges e_i are placed first on the cycle and each of the $W - w_i$ remaining edges is randomly placed in one of these w_i gaps. We recall the balls and bins results:

Lemma 5. *[10, 18] Throwing R balls independently and uniformly at random into Z bins. If $R = \Omega(Z \log Z)$, then the maximum number of balls in one bin is $O(R/Z)$ with probability $1 - O(1/R)$; if $R = o(Z \log Z)$, then the maximum load of bins is $O(\frac{\log Z}{\log\log Z})$ with probability $1 - O(1/R)$.*

If $W = \Omega(w_i \log w_i)$, the maximum gap among these w_i gaps is bounded by $O(W/w_i)$, with high probability in W. The number of buckets in this gap is $O(\frac{W}{bw_i}) = O(\frac{\Delta_p^2}{w_i})$. w.h.p. in W. Since each bucket uses a constant number of slots in expectation and the maximum load (number of buckets in the maximum gap) is highly concentrated around its mean in the balls and bins setting, we can directly multiply these two expected values to obtain the expected value for the refresh time for edge e_i. That is, the weighted refresh time for e_i is bounded by $w_i T_i = w_i \cdot O(\frac{\Delta_p^2}{w_i}) = O(\Delta_p^2)$ in expectation.

If $W = o(w_i \log w_i)$, a similar argument shows that the weighted refresh time for e_i is bounded by $O(\frac{w_i \log w_i}{b \log\log w_i}) = O(\frac{\Delta_p^2 \log w_i}{\log\log w_i})$ in expectation. Now compared to the lower bound of the optimal solution as in Theorem 4, our algorithm has an approximation factor of $O(\frac{\Delta_p \log m \log W_{\max}}{\log\log W_{\max}})$ in expectation. □

When the number of channels k is not 1, we will change the size of the buckets b to $\min\{\sqrt{\Delta_p}, \sqrt{k}\}W/\Delta_p^2$. The analysis is similar but a bit more technical.

Theorem 6. *Suppose we have k channels. Take $b = \min\{\sqrt{\Delta_p}, \sqrt{k}\}W/\Delta_p^2$, then the algorithm in Section 4.2 has an expected approximation factor of*

$$O\left(\max\left\{\sqrt{\Delta_p}, \frac{\Delta_p}{\sqrt{k}}\right\} \log m \log W_{\max} / \log\log W_{\max}\right).$$

Proof. Given k channels, observe that Theorem 4 is for general k, i.e., $\frac{\Delta_p}{7 \log m}$ gives a lower bound of the optimal solution. On the other hand, Lemma 4 also holds for general k and $b = \min\{\sqrt{\Delta_p}, \sqrt{k}\}W/\Delta_p^2$. For convenience, let $h = \min\{\sqrt{\Delta_p}, \sqrt{k}\}$. Hence, $b = hW/\Delta_p^2$.

$$\begin{aligned}\text{Prob}\{e_j \in G_i\} &\le b/W + (1 - (1 - b/W)^{\Delta_p})^2 \\ &\le h/\Delta_p^2 + (1 - e^{-h/\Delta_p})^2 \\ &\le h/\Delta_p^2 + h^2/\Delta_p^2 \le 2h^2/\Delta_p^2 \text{ since } h \ge 1.\end{aligned}$$

$$E[\Delta_i(e_j)] \le (w_j - 1 + 2(\Delta_p - w_j)) \cdot 2h^2/\Delta_p^2 \le 4h^2/\Delta_p$$

Similarly, we can get $E[\Delta_i'(e_j)] \le 2\Delta_p^2 \cdot 2h^2/\Delta_p^2 = 4h^2$.

By the linearity of expectation and the key idea of the proof of Theorem 3, the number of slots for bucket i, in expectation, is bounded by $O(4h^2/\Delta_p + 4h^2/k)$. By $h = \min\{\sqrt{\Delta_p}, \sqrt{k}\}$, $O(4h^2/\Delta_p + 4h^2/k) = O(8)$.

Next, let us analyze the number of edges in the gaps of edge e_i. If $W = \Omega(w_i \log w_i)$, the number of bins in this gap is at most $O\left(\frac{W}{bw_i}\right) = O\left(\frac{\Delta_p^2}{hw_i}\right)$ w.h.p. in W. Because of the high concentration around its mean in the balls and bins problem, we can calculate the expectation of the weighted refresh time directly by multiplying these two values mentioned earlier. That is, in expectation,

$$w_i T_i = w_i O\left(\frac{\Delta_p^2}{hw_i}\right) O(4h^2/\Delta_p + 4h^2/k) = O(h(\Delta_p + \Delta_p^2/k)).$$

Similarly, if $W = o(w_i \log w_i)$, the weighted refresh time in expectation for e_i is at most

$$\begin{aligned}w_i T_i &= w_i O\left(\frac{\log w_i}{b \log\log w_i}\right) O(4h^2/\Delta_p + 4h^2/k) \\ &= O\left(\frac{w_i \Delta_p^2 \log w_i}{Xh \log\log w_i}\right) O(4h^2/\Delta_p + 4h^2/k) \\ &= O\left(\frac{\log w_i}{\log\log w_i} h(\Delta_p + \Delta_p^2/k)\right)\end{aligned}$$

Combining these two upper bounds, in expectation, its approximation factor is bounded by $O(\min\{\sqrt{\Delta_p}, \sqrt{k}\} \cdot \Delta_p \log m \log W_{\max}/k \log\log W_{\max})$. Then by case analysis, we get the approximation factor as Theorem 6 states. □

5 Min Max (Weighted) Refresh Time in the Multi-Antenna Setting

In this section, we discuss the multi-antenna case. Since each node has $r > 1$ radios, it now can have at most r adjacent edges that are active in a time slot. The interference rule is the same as in the single-antenna setting: in a time slot, if two active edges are within interference range, then they must use different channels. The only difference is that now j adjacent edges incident to the same vertex v can be active in the same slot if they use different channels and different radio interfaces on v. The problem becomes more complicated than before. Fortunately, our algorithms for the weighted and unweighted problems can be generalized to the multi-antenna case.

Considering these two problems in the multi-antenna scenario, we have the following bounds for the optimal solution and the approximation factor for our scheduling algorithm.

Lemma 6. *In the multi-antenna case, the optimal solution has a maximum refresh time of at least* $\left\lceil \frac{2\Delta-1}{\min\{2r,k\}} \right\rceil$.

Proof. In the multi-antenna scenario, given an arbitrary edge e, at most $\min\{2r, k\}$ edges that are incident to e can appear in the same time slot. The reason is that for each endpoint of e, it can transmit at most $\min\{r, k\}$ messages successfully at the same time. There are at most $2\min\{r, k\}$ possible transmissions in total. They are within the interference area of e so they must use different channels. Hence, the maximum number of successful transmissions in a time slot is $\min\{k, 2\min\{r, k\}\}$, which is equivalent to $\min\{2r, k\}$. Remember that there are at most $2\Delta - 1$ edges in the interference range of edge e, so the maximum refresh time of the optimal solution is bounded from below by $\left\lceil \frac{2\Delta-1}{\min\{2r,k\}} \right\rceil$. □

Theorem 7. *In the multi-antenna case, the greedy algorithm gives a schedule of maximum refresh time at most* $\left\lceil \frac{2(\Delta-1)^2}{k} \right\rceil + \left\lceil \frac{2(\Delta-1)}{\min\{r,k\}} \right\rceil + 1$. *Therefore, this solution for the Min Max Refresh Time Scheduling problem is a* $\left\lceil \frac{\min\{2r,k\}(\Delta-1)}{k} \right\rceil + 2-$*approximation.*

Proof. It is similar to the proof of Theorem 3. The event "an edge incident to one endpoint of e is scheduled so we cannot schedule e" now becomes "we run out of channels or radios for one endpoint of e". That means for *each* of the r radios, an edge e' that is adjacent to e is scheduled with this radios or for *each* of the k channels, e' is scheduled with this channel. For this event to happen, we need to use at least $\min\{r, k\}$ such edges. Hence, there are at most $\left\lceil \frac{2(\Delta-1)}{\min\{r,k\}} \right\rceil$ slots of this type. The maximum refresh time now is at most $\left\lceil \frac{2(\Delta-1)^2}{k} \right\rceil + \left\lceil \frac{2(\Delta-1)}{\min\{r,k\}} \right\rceil + 1$. Furthermore, the lower bound of optimal max refresh time $\left\lceil \frac{2\Delta-1}{\min\{2r,k\}} \right\rceil$ implies that this greedy algorithm produces a $\left\lceil \frac{\min\{2r,k\}(\Delta-1)}{k} \right\rceil + 2$-approximation of the optimal. □

Lemma 7. *In the multi-antenna case, the optimal solution for the Min Max Weighted Refresh Time Scheduling problem has* $\mathbb{O}(S^*) \geq \frac{\Delta_p}{7\log m \min\{r,k\}}$.

Proof. First of all, notice that Lemma 1 and Lemma 2 still hold in the multi-antenna scenario. Only the proof of Lemma 1 is different. That is inserting e_i may remove more than one occurrence for any e_j, but no more than $\min\{r,k\}$ occurrences. With at most $\min\{r,k\}$ occurrences, we still can give a sufficiently large D that yields contradiction. Now we get an inequality, $T(S) \geq \Delta_p / \min\{r,k\}$. Combining with Lemma 2, we get $\frac{\Delta_p}{7 \log m \min\{r,k\}}$ as a lower bound for optimal.

Before ending this proof, let us clarify why an edge now can appear at most $\min\{r,k\}$ times in the same time slot. It is because that for edges that have common endpoints, if we require these edges appear in the same time slot, then they must use different radios with different channels. Different radios represent different interfaces in the endpoints and different channels mean no interference occurs. Hence, there are at most $\min\{r,k\}$ duplicate edges in a time slot. □

Theorem 8. *In the multi-antenna case with $b = \min\{\sqrt{\Delta_p \min\{r,k\}}, \sqrt{k}\} W_{sum}/\Delta_p^2$, our algorithm for the Min Max Weighted Refresh Time Scheduling problem has an expected approximation factor*

- *$O(\Delta_p \sqrt{k} \cdot \log m \log W_{max} / \log\log W_{max})$, if $k \leq r$;*
- *$O(r\Delta_p/\sqrt{k} \cdot \log m \log W_{max} / \log\log W_{max})$, if $k > r$ and $k \leq r\Delta_p$;*
- *$O(\sqrt{r\Delta_p} \cdot \log m \log W_{max} / \log\log W_{max})$, if $k > r\Delta_p$.*

Proof. In the multi-antenna setting with k channels, Lemma 7 still holds. That is, $\frac{\Delta_p}{7 \log m \min\{r,k\}}$ serves as a lower bound for the optimal solution. On the other hand, Lemma 4 also holds for general k and $b = \min\{\sqrt{\Delta_p \min\{r,k\}}, \sqrt{k}\} W_{\text{sum}}/\Delta_p^2$. The only difference is the following. By the linearity of expectation and the key idea of the proof of Theorem 7, the number of slots for bucket i, in expectation, is bounded by $O\left(\frac{4h^2}{\Delta_p \min\{r,k\}} + \frac{4h^2}{k}\right)$. Using $h = \min\left\{\sqrt{\Delta_p \min\{r,k\}}, \sqrt{k}\right\}$, we get

$$O\left(\frac{4h^2}{\Delta_p \min\{r,k\}} + \frac{4h^2}{k}\right) = O(8).$$

The analysis for the number of edges in gaps of edge e_i remains the same. Hence, when $W_{\text{sum}} = \Omega(w_i \log w_i)$, in expectation,

$$w_i T_i = w_i O\left(\frac{\Delta_p^2}{h w_i}\right) O\left(\frac{4h^2}{\Delta_p \min\{r,k\}} + \frac{4h^2}{k}\right) = O\left(h\left(\frac{\Delta_p}{\min\{r,k\}} + \frac{\Delta_p^2}{k}\right)\right).$$

Similarly, when $W_{\text{sum}} = o(w_i \log w_i)$, $w_i T_i = O\left(\frac{\log w_i}{\log\log w_i} h \left(\frac{\Delta_p}{\min\{r,k\}} + \frac{\Delta_p^2}{k}\right)\right)$.

As a result, the expected upper bound is

$$w_i T_i = O\left(\frac{\log W_{\max}}{\log\log W_{\max}} h \left(\frac{\Delta_p}{\min\{r,k\}} + \frac{\Delta_p^2}{k}\right)\right).$$

Its approximation factor is bounded from above by

$$O\left(\frac{\log W_{\max}}{\log\log W_{\max}} \min\{\sqrt{\Delta_p \min\{r,k\}}, \sqrt{k}\} \cdot \left(\frac{1}{\min\{r,k\}} + \frac{\Delta_p}{k}\right) \log m \min\{r,k\}\right).$$

Then by case analysis, we get the expected approximation factors, which is stated in Theorem 8. □

6 Conclusion and Future Work

There are a few directions to explore in future work. One direction is to consider the physical model (SINR model) as our interference model. Another direction is to generalize the problem. In general, the problem of fair scheduling with conflicting constraints (low delay and interference) goes beyond scheduling wireless links. We expect our techniques can be applied to a broader setting.

Acknowledgements: This work was supported in part by NSF grants CCF-1439084, CCF-1535900, CNS-1553510, CNS-1618391, CNS-1553273, and DMS-1737812.

References

1. Al Islam, A.A., Islam, M.J., Nurain, N., Raghunathan, V.: Channel assignment techniques for multi-radio wireless mesh networks: A survey. IEEE Communications Surveys & Tutorials **18**(2), 988–1017 (2015)
2. Balakrishnan, H., Barrett, C.L., Kumar, V.S., Marathe, M.V., Thite, S.: The distance-2 matching problem and its relationship to the mac-layer capacity of ad hoc wireless networks. IEEE J.Sel. A. Commun. **22**(6), 1069–1079 (Sep 2006)
3. Barrett, C.L., Istrate, G., Kumar, V.S.A., Marathe, M.V., Thite, S., Thulasidasan, S.: Strong edge coloring for channel assignment in wireless radio networks. In: Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06). pp. 5 pp.–110 (March 2006)
4. Bicket, J., Aguayo, D., Biswas, S., Morris, R.: Architecture and evaluation of an unplanned 802.11 b mesh network. In: Proceedings of the 11th annual international conference on Mobile computing and networking. pp. 31–42. ACM (2005)
5. CHAMBERS, B.: A rooftop ad hoc wireless network. http://www. pdos. lcs. mit. edu/grid/ (2002)
6. Chaporkar, P., Kar, K., Luo, X., Sarkar, S.: Throughput and fairness guarantees through maximal scheduling in wireless networks. IEEE Transactions on Information Theory **54**(2), 572–594 (Feb 2008)
7. Committee, L.S., et al.: Wireless lan medium access control (mac) and physical layer (phy) specifications: High-speed physical layer in the 5 ghz band. Piscataway, NJ: IEEE Std **802**(1) (1999)
8. CREATE-NET, Technion: Wing: Wireless mesh network for next-generation internet. http://www.wingproject. org (2012), accessed 15 Jun. 2019
9. Ghosh, A., Wolter, D.R., Andrews, J.G., Chen, R.: Broadband wireless access with wimax/802.16: current performance benchmarks and future potential. IEEE communications magazine **43**(2), 129–136 (2005)
10. Gonnet, G.H.: Expected length of the longest probe sequence in hash code searching. Journal of the ACM (JACM) **28**(2), 289–304 (1981)
11. Group, I..W., et al.: Part 11: wireless lan medium access control (mac) and physical layer (phy) specifications: higher-speed physical layer extension in the 2.4 ghz band. ANSI/IEEE Std 802.11 (1999)

12. Hajek, B., Sasaki, G.: Link scheduling in polynomial time. IEEE Transactions on Information Theory **34**(5), 910–917 (Sep 1988)
13. Holyer, I.: The NP-completeness of edge-coloring. SIAM Journal on Computing **10**(4), 718–720 (1981)
14. Joo, C., Lin, X., Shroff, N.B.: Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks. In: IEEE INFOCOM 2008 - The 27th Conference on Computer Communications. pp. 1777–1785 (April 2008)
15. Kumar, V.S.A., Marathe, M.V., Parthasarathy, S., Srinivasan, A.: End-to-end packet-scheduling in wireless ad-hoc networks. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1021–1030. SODA '04, Philadelphia, PA, USA (2004)
16. Mahdian, M.: The strong chromatic index of graphs. Dissertation, Department of Computer Science, University of Toronto (2000)
17. Nandagopal, T., Kim, T.E., Gao, X., Bharghavan, V.: Achieving mac layer fairness in wireless packet networks. In: Proceedings of the 6th annual international conference on Mobile computing and networking. pp. 87–98. ACM (2000)
18. Raab, M., Steger, A.: "balls into bins" – a simple and tight analysis. Randomization and Approximation Techniques in Computer Science pp. 159–170 (1998)
19. Ramanathan, S., Lloyd, E.L.: Scheduling algorithms for multihop radio networks. IEEE/ACM Transactions on Networking **1**(2), 166–177 (Apr 1993)
20. Ramanathan, S.: A unified framework and algorithm for channel assignment in wireless networks. Wireless Networks **5**(2), 81–94 (1999)
21. Ramanathan, S., Lloyd, E.L.: Scheduling algorithms for multihop radio networks. IEEE/ACM Transactions on Networking (TON) **1**(2), 166–177 (1993)
22. Sharma, G., Mazumdar, R.R., Shroff, N.B.: On the complexity of scheduling in wireless networks. In: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking. pp. 227–238. MobiCom '06, ACM, New York, NY, USA (2006)
23. SHI, H., Prasad, R.V., Onur, E., Niemegeers, I.G.M.M.: Fairness in wireless networks:issues, measures and challenges. IEEE Communications Surveys Tutorials **16**(1), 5–24 (First 2014)
24. Si, W., Selvakennedy, S., Zomaya, A.Y.: An overview of channel assignment methods for multi-radio multi-channel wireless mesh networks. Journal of Parallel and Distributed Computing **70**(5), 505–524 (2010)
25. Stockmeyer, L.J., Vazirani, V.V.: Np-completeness of some generalizations of the maximum matching problem. Inf. Process. Lett. **15**(1), 14–19 (1982)
26. Wan, P.J., Frieder, O., Jia, X., Yao, F., Xu, X., Tang, S.: Wireless link scheduling under physical interference model. IEEE (2011)
27. Wan, P.J., Jia, X., Dai, G., Du, H., Wan, Z., Frieder, O.: Scalable algorithms for wireless link schedulings in multi-channel multi-radio wireless networks. In: INFOCOM, 2013 Proceedings IEEE. pp. 2121–2129 (2013)
28. Wu, X., Srikant, R., Perkins, J.R.: Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks. IEEE Transactions on Mobile Computing **6**(6), 595–605 (June 2007)

A Appendix: Omitted Proof

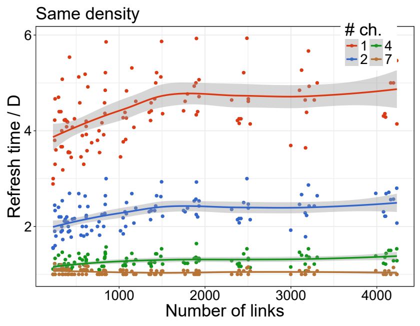
A.1 Proof on Schedule Periodicity

Given any infinite schedule S with a finite maximum refresh time, we can find a periodic schedule with max refresh time for each edge e_i no worse than that in S. Let the max refresh time of all edges in S be L. Consider the family of all possible schedules of the edges of G with no interference with length L. The number of these schedules is finite.

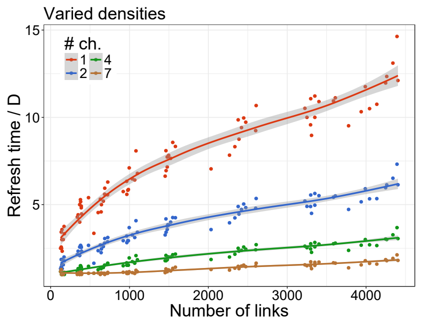
Now, let's construct a periodic schedule S' from S. We divide S into sub-schedules of length L each. Since the configuration of these sub-schedules of length L is finite and S is infinitely long, there exists a subschedule M that repeats at some point in S. Extract the subschedule of S that starts from the first appearance of M and ends right before the second appearance of M. We now repeat this sub-schedule periodically and call it S'.

Since each sub-schedule has length L, any edge e_i appears at least once in each sub-schedule. Thus, all the gap between two successive time slots of the same edge e_i in S' also happens in the original schedule S. Hence, the constructed periodic schedule has a maximum refresh time for each edge e_i which is no worse than that in the original schedule S. □

B Evaluation



(a) Same density

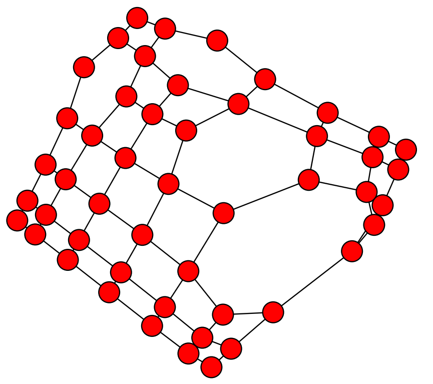


(b) Varied densities

Fig. 3. Unit disk network with random node placement. The node degree is kept similar or increases when the scale of network increases to thousand of edges.

In this section, we evaluate our unweighted and weighted channel assignment algorithms under different scenarios in the single antenna case. Without loss of generality, we can assume that the smallest weight is 1 and all other weights are rounded to integer values. We consider model networks such as random node placement and perturbed grid placement with unit disk communication capacity, and also a real testbed network (denoted the Tmote network) which consists of 48 TMotes in a building that uses the ChipCon CC2420 radio. We vary network parameters such as node degree, the number of channels, weight distributions and measure the performance of our algorithms using the maximum fresh time divided by maximum (weighted) degree and Δ_p as the metric. For each network, we ran our algorithm 50 times to compute the average performance.

The network topology in Figure 3 is constructed by throwing random nodes with a uniform circular range in a 2D unit square. This imitates random node placement in the wild. For each evaluation, we generate 50 networks. In Figure 3.a, we increase the number of nodes in the unit square from 50 to 600 while keeping the average degrees the same (by scaling down the communication range of each node), so every node continues to have a similar number of interfering counterparts even when the network scale increases. The almost flat slopes of curves indicate that our algorithm still works as efficiently for large graphs as for small graphs when those graphs have similar densities. Besides, the result shows that when we have a reasonable number of channels, our algorithm can efficiently assign channels to a large network while keeping the latency low. In Figure 3.b, we increase the number of nodes but keep the communication range the same, i.e., when the number of nodes increases, the network becomes denser. That means a lot of implicit interferences occur. Therefore, the maximum refresh time increases unavoidably. Still, when we have a reasonable number of channels, our algorithm can keep the max refresh time moderate.


(a) Unit disk network with perturbed grids

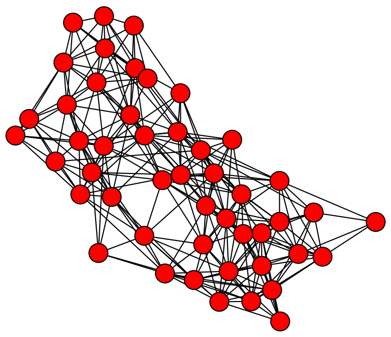

(b) Tmote network with 48 Nodes

Fig. 4. Visualization of the network topologies

Random node placement often leads to many small holes in the network. To make the network more robust, perturbed grid placement is preferred which gives a more stable node degree among the network while reducing the number of gaps inside. Therefore, we often see an almost grid placement in real-world sensor networks. In order to evaluate on such wireless networks, we use a perturbed 7×7 grid placement network shown in Figure 4.a and also a Tmote network as shown in Figure 4.b. In the Tmote network, these nodes are deployed on walls and ceilings of a building. We collect traces of 3,600,000 packet transmissions using IEEE 802.15.4 standard for each pair of nodes. With the transmission traces, we define two nodes are connected if and only if the packet reception rate of its link is over 90%.

In Figure 5, we evaluate our algorithm on these two networks when weight distributions are uniform and power-law. In both networks, our algorithm can efficiently

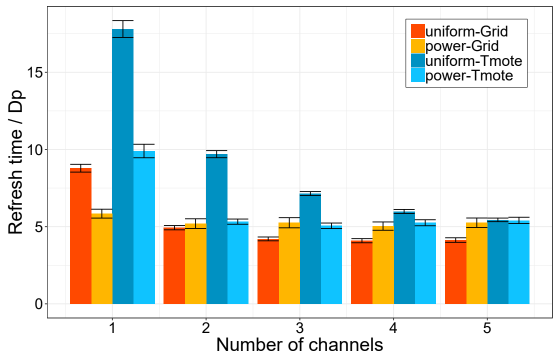


Fig. 5. Performance of weighted channel assignment on perturb grid and Tmote network with varying number of channels and weight distributions.

use channels to reduce the refresh time. However, when the weight distribution is power-law, the benefit diminishes because some implicit interference is unavoidable. Note that some edges have very high priorities and they contribute to the weighted degree which makes the maximum weighted degree pretty high. The ratio of the maximum weighted degree to the total weight is 0.32 for the perturbed grid and it is 0.12 for the Tmote network. The node with the maximum weighted degree creates more unavoidable interferences in the perturbed grid. Hence, the performance of the Tmote network is better than the one of the perturb grid. On the other hand, for both networks under the uniform distribution, the ratios are the same, 0.06, which is quite small and leaves room for improvement of our algorithm. When we vary the number of channels from one to two, our algorithm improves the most. It is almost twice as better than the case of only one channel.