# Scheduling Dynamic Wireless Networks with Limited Operations

Haoyang Lu and Wei Gao
Department of Electrical Engineering and Computer Science
University of Tennessee at Knoxville

***Abstract*—Scheduling in wireless networks is critical to maximize the network throughput by avoiding interference among wireless links, and is usually formulated as solving the NP-hard Maximum Weighted Independent Set (MWIS) problem over a network conflict graph. Existing scheduling algorithms are designed to provide approximations to global optimality via distributed operations in wireless networks, but will frequently reschedule the entire network in cases of network dynamics regardless of the actual network area being affected by these dynamics. Such repetitive rescheduling results in a large amount of computation and communication overhead, most of which may be, however, unnecessarily incurred over the wireless links that remain unchanged. To reduce such overhead and improve the scheduling cost-effectiveness, in this paper we develop distributed algorithms that adaptively constrain network scheduling within the limited scope where network dynamics occur. The scheduling results from such limited operations are then combined with the previous scheduling results over the remaining portions of the network, hence still providing guaranteed network throughput. The performance of our proposed algorithms has been validated by formal analysis, and been verified by both numerical studies and real-world experiments.**

## I. Introduction

In multi-hop wireless networks, a node may have multiple wireless links connecting other nodes, but simultaneous data transmissions over these links may fail due to interference. Appropriate link scheduling, hence, is crucial to maximize the wireless network throughput [18]. Especially in emerging cognitive-radio networks [22] and next-generation cellular networks [14] where heterogeneous types of wireless devices share the channel, efficient network scheduling algorithms are indispensable to efficient utilization of wireless spectrum.

A lot of research has been done on developing scheduling algorithms that maximize the wireless network throughput, and most existing algorithms formulate the scheduling problem to finding the Maximum Weighted Independent Set (MWIS) of a conflict graph, in which each vertex corresponds to a wireless link and each edge connects two vertices (links) that interfere with each other. The wireless links corresponding to the vertices in the MWIS, then, can transmit data simultaneously and maximize the network throughput if their weights represent the contribution of the corresponding links on the network throughput. Since finding the MWIS in a graph is generally a NP-hard problem, optimal algorithms such as MaxWeight [21] incur large amounts of computational overhead and only approximation methods are feasible in practical wireless networks [26], [24]. Thereafter, researchers have been focusing on developing different types of distributed scheduling algorithms [3], [11], [9], [17], [20], [16]. These algorithms trade the global optimality of network throughput for reduced scheduling overhead, and are practically deployable in wireless networks via message exchange between wireless devices that maintains the local information about link status.

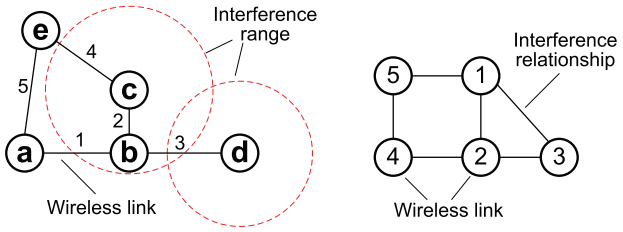
All the existing network scheduling algorithms are designed with respect to a fixed network conflict graph, which may, however, change in practical wireless networks due to network dynamics. Such network dynamics may include *i)* heterogeneous levels of wireless channel fading that lead to frequent changes of link data rates, and *ii)* fluctuating application traffic patterns that result in unstable packet queue lengths at wireless devices. In these cases, the existing algorithms will reschedule the entire network as long as the status of any wireless link changes, no matter whether such dynamics actually impact every link in the network. Particularly when the quality of the wireless channel is degraded or unstable, such rescheduling will be operated at a high frequency (e.g., once per second) at the global network scope, resulting in large amounts of computation and communication overhead.

Instead, by experimentally investigating wireless network dynamics in different types of practical application scenarios, we observe that these dynamics in wireless networks usually affect only a small portion of the network at any specific moment, when the rest major portion of the network remains stable and unaffected. Based on this observation, in this paper we develop distributed algorithms that schedule wireless networks only within the limited scope where network dynamics occur. Our proposed algorithms combine the scheduling results from such limited operations with the previous scheduling results over the remaining portions of the network, hence still providing guaranteed network throughput with much lower overhead. Our detailed contributions are as follows:

- To the best of our knowledge, we are the first who constrain wireless network scheduling to a limited scope, dramatically reducing the scheduling overhead in practice without noticeably impairing the network throughput.
- We conducted formal analysis that demonstrates the correctness of our scheduling algorithms on avoiding network interference, and also shows that our algorithms provide a fixed approximation ratio to optimality.
- We evaluated the performance and overhead of our scheduling algorithms using both numerical studies with large-scale conflict graphs and real-world experiments over software-defined radio platforms that take practical

This work was supported in part by the US National Science Foundation (NSF) under grant number CNS-1456656, CNS-1526769 and CNS-1553395. This work was also supported in part by the US Army Research Office (ARO) under grant number W911NF-15-1- 0221.

**978-1-5090-3281-5/16/$31.00 ©2016 IEEE**




(a) Protocol interference model (b) Conflict graph model

Fig. 1: Illustration of system model. Note that the interference ranges of different devices may be different.

wireless channel conditions into account.

The rest of this paper is organized as follows. Section II reviews the related work. Section III provides a high-level overview of our approach. Section IV presents the details of scheduling algorithms and Section V formally analyzes these algorithms. Section VI and Section VII evaluate our algorithms via numerical studies and real-world experimentation, respectively. Section VIII finally concludes the paper.

## II. Related Work

Research on wireless network scheduling began with the development of backpressure routing protocols, which maximizes the throughput in a queuing network by appropriately directing traffic through multiple network nodes and can be considered as an extension of max-weight scheduling [4]. Although the MaxWeight routing algorithm [21] has been well known to be throughput optimal by stabilizing the queues and traffic rates in the network capacity region, it has a high computational complexity. Instead, various approximation methods have been proposed to guarantee a non-negligible fraction of the optimal network throughput with lower overhead, by converting the scheduling problem to finding the MWIS from the network conflict graph and assuming ideal non-fading wireless channels [9]. The Greedy Maximal Scheduling (GMS) algorithm reaches a provable fraction of the optimal throughput with a linear complexity [1], and Maximal Scheduling algorithm further reduces the complexity to $O(\text{Log}\, N)$ [25]. Other algorithms [2], [8], [15] achieve the optimal network throughput with linear or constant complexities, but lead to poor delay performance. The complexity of wireless network scheduling has also been studied from a theoretical aspect [18], [26], which provides analytical insights to algorithm designs.

Early-stage scheduling algorithms require global information about wireless network links and are hence not amenable for distributed implementation in practical networks. Later studies have been focusing on distributed scheduling algorithms that require only local information about wireless links at each wireless device, and have also taken the dynamic characteristics of channel fading [12] and physical interference [23] into account. Many algorithms adopt greedy heuristics as the scheduling strategy and achieve a fixed fraction of the optimal throughput with fading channels [9], [17]. Other algorithms employ various types of tools and techniques such as graph theory [3], LP-relaxation [20] and message passing [16] to further improve the approximation ratio based on certain assumptions on the conflict graph topology, e.g., bipartite graphs. However, all these algorithms are designed for a given and fixed conflict graph. When the topology or vertex weights of the conflict graph change, the algorithm has to be re-operated over the entire network. In contrast, our proposed algorithms can provide provable network throughput by only scheduling the network within a limited scope.

TABLE I: Notation Summary

| Notation | Explanation |
|---|---|
| $G$ | The network conflict graph |
| $V$ | The set of vertices in $G$ |
| $E$ | The set of edges in $G$ |
| $\mathbb{C}$ | The collections of Independent Sets (ISs) of $G$ |
| $\mathcal{S}_t$ | The MWIS of $G$ found by the scheduling algorithm at time $t$ |
| $w_{v,t}$ | The weight of vertex $v$ in $G$ at time $t$ |
| $\mathcal{V}_t$ | The Variation Set of $G$ at time $t$ |
| $\mathcal{O}_t$ | The Operating Set of $G$ at time $t$ |
| $\mathcal{O}_{i,t}$ | $\mathcal{O}_t$ when only vertex $i$ on the conflict graph changes its weight |
| $N(v)$ | The set of neighbors of vertex $v$ on the conflict graph |
| $d(u,v)$ | The length of the shortest path between vertices $u$ and $v$ |
| $W(V)$ | The sum of vertex weights in the vertex set $V$ |

## III. Overview

In this section, we present the high-level overview of our scheduling approach, with respect to the notations in Table I.

### A. System Model

We consider a generic scenario of a multi-hop wireless network, where wireless devices connect with each other via direct links, and these links share a single wireless channel. Concurrent data transmissions over different links, hence, may conflict and fail due to wireless interference. Without loss of generality, we adopt the protocol interference model [23], which is also known as the unified disk graph model in a multi-hop wireless network, to depict such interference relationship between wireless links. This model differentiates a wireless device's interference range, which relates to multiple aspects of the channel's physical characteristics [19], from the device's communication range that is mainly determined by the channel signal-to-noise-ratio (SNR). Hence, it extends the traditional model based on the signal to interference and noise ratio (SINR) [5] and incorporates the interference characteristics of heterogeneous networks such as CDMA and 802.11.

Based on this model, the transmission between two devices $a$ and $b$ only succeeds if *i)* the two devices are within the communication range of each other and connected by a wireless link, and *ii)* $b$ is outside the interference range of any other transmitter at the moment. Otherwise, wireless links between $b$ and all other transmitters within the interference range of $b$ are considered to *conflict* with each other. For example in Figure 1(a), link 1 will conflict with link 2 if they transmit data simultaneously, because $b$ is in the interference range of $c$. Note that links 1 and 3 are also conflicting because $b$ cannot send and receive data at the same time.

This model results in a *conflict graph* $G = (V, E)$ that depicts the interference relationship among wireless links for a given network topology. In the conflict graph, each wireless link is mapped onto a vertex $v \in V$, and the weight associated with each vertex is usually defined as the product of the link's data rate and packet queue length [9]. Two wireless links that interfere with each other, then, are connected in the conflict

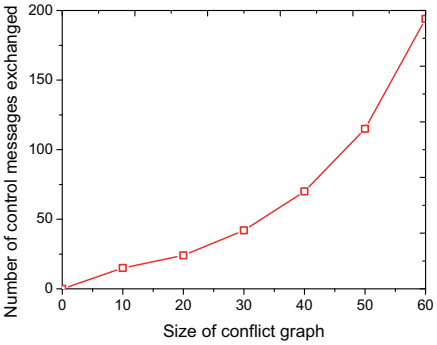



Fig. 2: Average number of control message exchanges for executing the DistGreedy algorithm [9] once

graph by an edge in $E$. For example, Figure 1(b) shows the conflict graph for the wireless network topology in Figure 1(a).

The above system model allows us to formulate the wireless network scheduling problem as finding the MWIS of $G$, which indicates the set of links that can transmit data simultaneously while maximizing the network throughput[1]. More specifically, by dividing time into equal slots and letting $\mathbb{C}$ be the collection of all the feasible independent sets (ISs)[2] of $G$ at time $i$, the scheduling problem is formulated as finding $\mathcal{S}_t$, such that

$$\mathcal{S}_t = \arg\max_{\mathcal{S}\in\mathbb{C}} \sum\nolimits_{j\in\mathcal{S}} w_{j,t}.$$

where $w_{j,t}$ is the vertex weight of node $j$ at slot $t$.

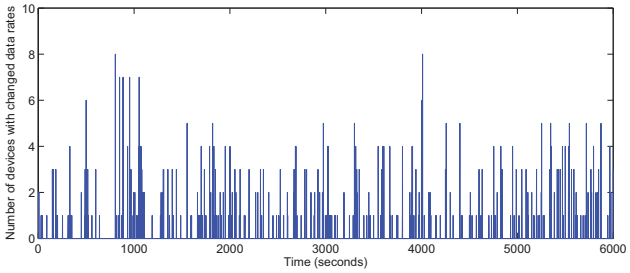
### B. Motivation

In practice, due to various types of dynamics in a wireless network, the vertex weights on the conflict graph may dynamically change over time. To maximize the network throughput in these cases, existing network scheduling algorithms opt to reschedule the entire network as soon as the weight of any vertex has changed. However, such frequent rescheduling over the global network scope will incur a large amount of communication and computation overhead, especially when the network size increases and the network scheduling algorithm is operated in a fully distributed manner. For example, Figure 2 shows that when the size of conflict graph grows from 10 to 60, the amount of control messages being exchanged among mobile nodes by running the DistGreedy scheduling algorithm [9] dramatically increases by more than 10 times. Considering that the changes of wireless link states may be highly frequent in practice, the scalability of these existing scheduling algorithms is seriously limited.

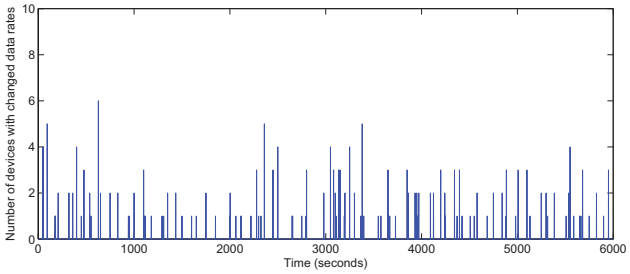
Instead, our proposed scheduling approach is motivated by the observation that wireless network dynamics, which lead to fluctuations in both link data rates and packet queue lengths at wireless devices, usually occur over only a small portion of these devices in the network at any moment. First, the fading characteristics of wireless channels are highly location-dependent, and it is hence unlikely for multiple wireless

[1]The network throughput is represented by the weights of vertices in the MWIS of the conflict graph.

[2]An independent set (IS) of a graph is a set of graph vertices, among which no two are adjacent.




(a) Small office room




(b) Large indoor space

Fig. 3: Preliminary experiment results: Data rate variations of multiple wireless links in practical wireless networks

devices at different locations to experience the same level of channel fading and have their link data rates changed simultaneously. To verify this observation, we conducted a preliminary experiment by connecting 10 mobile devices (4 laptops, 4 smartphones and 2 tablets) to a wireless Access Point (AP) in different indoor scenarios and observing the data rate variations of their wireless links to the AP over 100 minutes[3]. The experiment results in Figure 3 show that even in a small office room with strong channel fading and multi-path effects, the amount of wireless devices that have simultaneously changed their link data rates will not exceed 20% in most cases. Second, the packet queue length of a wireless device is mainly determined by the network traffic pattern generated from user applications. Since the behavior and application executions of each mobile user are independent from others, the packet queue lengths at multiple devices are also unlikely to change simultaneously.

As described in Section III-A, the vertex weights in the conflict graph are jointly determined by the data rate and packet queue lengths at wireless devices. Hence, the two observations above highlight that at any time slot, it is only necessary to reschedule a limited portion of the network, where the link characteristics and the corresponding vertex weights on the conflict graph actually change. Furthermore, many existing wireless network scheduling algorithms, such as Q-CSMA [15], adopt log functions during the scheduling process, leading to a limited amount of vertex weight changes on the conflict graph. As a result, although the network still needs to be frequently rescheduled, such rescheduling with limited operations incurs negligible overhead at each time.

[3]The Minstrel algorithm [13] is used to adapt the data rate of each wireless link to the up-to-date link condition.

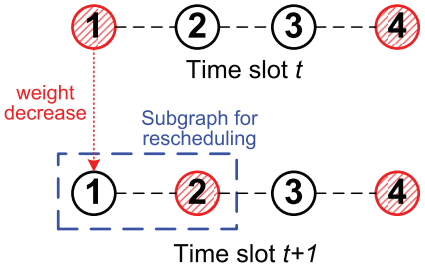



Fig. 4: Wireless scheduling with limited operations

### C. The Big Picture

Motivated by the above observations, our approach only reschedules the limited portion of the network where network dynamics occur at each time slot. The scheduling result from such limited operations is then combined with the scheduling results over the remaining components of the network at previous time slots, and improves the cost-effectiveness of network scheduling by reusing these previous results.

The big picture of our approach is illustrated by an example in Figure 4, which shows a conflict graph with four vertices. At time slot $t$, the MWIS of the conflict graph is $\{1, 4\}$. Then, when the weight of vertex 1 decreases at time slot $t+1$, instead of rescheduling over the entire conflict graph, we only re-find the MWIS in a limited subgraph that contains vertex 1 and its 1-hop neighbor (vertex 2). In Figure 4, if $w_{2,t+1} > w_{1,t+1}$, the MWIS of this subgraph will contain only vertex 2 and hence the new scheduling result at time slot $t+1$ will be $\{2, 4\}$.

The major difficulty of such network scheduling, however, is two-fold. First, the scheduling algorithm has to ensure that the scheduling result of the limited operations, when combined with the previous scheduling results over the remaining part of the network, is still an Independent Set (IS) of the conflict graph and does not cause any interference when being applied for wireless data transmissions. Essentially, this characteristic implies that the union of two ISs still produces an IS, and we will develop distributed algorithms in Section IV to address the possible conflict between the two ISs when they are being combined. Second, the combined IS of the conflict graph should still provide high throughput over the entire wireless network. Intuitively, the larger portion of the network is rescheduled, the closer the scheduling result approximates to optimality. We will formally investigate the performance approximation ratio provided by our scheduling algorithm in Section V, and then explore the tradeoff between throughput optimality and scheduling overhead via both numerical studies and real-world experiments in Sections VI and VII.

## IV. Scheduling Algorithm

In this section, we present our algorithms of finding the MWIS of the conflict graph with limited operations, when multiple vertices of the graph change their weights due to network dynamics. We define the set of vertices whose weights change at time slot $t$ as the ***Variation Set*** $\mathcal{V}_t$, and the set of vertices we operate at $t$ as the ***Operating Set*** $\mathcal{O}_t$.

Generally speaking, compared with traditional scheduling approaches that find the MWIS of the complete conflict graph, our approach reduces the network rescheduling overhead by finding the MWIS within the $k$-hop neighborhood of $\mathcal{V}_t$. Hence, $\mathcal{O}_t$ could intuitively be the union of the $k$-hop neighborhoods of all vertices in $\mathcal{V}_t$. However, we exploit two exceptions to further reduce the size of $\mathcal{O}_t$: *1)* some vertices in $\mathcal{V}_t$ may not need to be included in $\mathcal{O}_t$; and *2)* some vertices in the $k$-hop neighborhood of $v \in \mathcal{V}_t$ may not need to be included in $\mathcal{O}_t$. Afterwards, we adopt existing MWIS algorithms to find the MWIS over $\mathcal{O}_t$ and combine it with the previous scheduling result. Without loss of generality, in the rest of this section we call $\mathcal{S}_t$ produced by our scheduling algorithm as the MWIS of the corresponding vertex set at $t$, although it may not be the optimal solution at the global scope. In Section V, we will further analyze the performance of our scheduling algorithm in terms of its approximation ratio to optimality.

### A. Necessity of Rescheduling

To efficiently decide the minimum Operating Set, we first provide analytical insights on whether a vertex in the Variation Set needs to be involved for rescheduling. We consider a generic scenario, where the MWIS ($\mathcal{S}_{t-1}$) of the conflict graph $G$ at the previous time slot $t-1$ has been known and the weights of multiple vertices in $G$ change at time slot $t$. For any two vertices $u$ and $v$, we denote the set of their neighbors at $N(u)$ and $N(v)$, respectively, and the length of the shortest path between them as $d(u, v)$. For any vertex set $V$, we denote $W(V) = \sum_{v \in V} w_v$ as the sum of vertex weights within $V$.

We first analyze the Variation Set that only contains one vertex, i.e., $\mathcal{V}_t = \{v\}$, and we have the following lemmas.

**Lemma 1.** *When $\mathcal{V}_t = \{v\}$, if vertex $v$ satisfies* i) $w_{v,t} > w_{v,t-1}$ *and* ii) $v \in \mathcal{S}_{t-1}$, *then we have $\mathcal{S}_t = \mathcal{S}_{t-1}$.*

*Proof:* Lemma 1 can be proved by finding a contradiction. Suppose that $\mathcal{S}_t \neq \mathcal{S}_{t-1}$, then there are two cases for vertex $v$ and we discuss them separately.

*1)* $v \in \mathcal{S}_t$: Since $\mathcal{S}_{t-1} \neq \mathcal{S}_t$, we have $\mathcal{S}_{t-1} \setminus \{v\} \neq \mathcal{S}_t \setminus \{v\}$. Because $\mathcal{S}_t$ is the MWIS of $G$ at slot $t$, we have

$$w_{v,t} + W(\mathcal{S}_t \setminus \{v\}) > w_{v,t} + W(\mathcal{S}_{t-1} \setminus \{v\}).$$

Therefore, we have

$$w_{v,t-1} + W(\mathcal{S}_t \setminus \{v\}) > w_{v,t-1} + W(\mathcal{S}_{t-1} \setminus \{v\}),$$

which indicates that $\{v\} \cup \{\mathcal{S}_t \setminus \{v\}\}$ is the MWIS of $G$ at slot $t-1$. This fact, however, contradicts with that $\mathcal{S}_{t-1}$ is the MWIS of $G$ at slot $t-1$, where $\mathcal{S}_{t-1} = \{v\} \cup \{\mathcal{S}_{t-1} \setminus \{v\}\}$.

*2)* $v \notin \mathcal{S}_t$: In this case, $\mathcal{S}_t \subset V \setminus \{v\}$. therefore,

$$\begin{aligned} W(\mathcal{S}_t) &> w_{v,t} + W(\mathcal{S}_{t-1} \setminus \{v\}) > w_{v,t-1} + W(\mathcal{S}_{t-1} \setminus \{v\}) \\ &= W(\mathcal{S}_{t-1}), \end{aligned}$$

which indicates that $\mathcal{S}_t$ is also the MWIS of $G$ at time slot $t-1$. This fact, however, contradicts with that $\mathcal{S}_{t-1}$ is the MWIS at slot $t-1$ and $\mathcal{S}_t \neq \mathcal{S}_{t-1}$.

This lemma is then proved by combining the conclusions of the two cases above. ■

Similarly, we have the following lemma:

**Lemma 2.** *When $\mathcal{V}_t = \{v\}$, if vertex $v$ satisfies* i) $w_{v,t} < w_{v,t-1}$ *and* ii) $v \notin \mathcal{S}_{t-1}$, *then we have $\mathcal{S}_t = \mathcal{S}_{t-1}$.*

*Proof:* Lemma 2 could be proved by finding a contradiction following the same way as Lemma 1 is proved. ■

Based on Lemma 1 and Lemma 2, we have the following theorem for any $\mathcal{V}_t$ with arbitrary sizes:

**Theorem 1.** *For any $v \in \mathcal{V}_t$, $v \notin \mathcal{O}_t$ if any of the two conditions is satisfied:* i) $w_{v,t} > w_{v,t-1}$ *and* $v \in \mathcal{S}_{t-1}$ *or* ii) $w_{v,t} < w_{v,t-1}$ *and* $v \notin \mathcal{S}_{t-1}$.

*Proof:* For $\mathcal{V}_t = \{v_1, v_2, ..., v_n\}$, Theorem 1 can be proved by recursively applying Lemma 1 and Lemma 2 to $v_i$ and $\mathcal{V}_t \setminus \{v_i\}$. ■

Theorem 1 highlights that some vertices in the Variation Set do not need to be involved into the Operating Set, no matter how their weights are changed at the current time slot. These vertices, hence, are also excluded from the network rescheduling process to reduce the rescheduling overhead. Without loss of generality, in the rest of this paper we use a binary variable $I_{v,t} \in \{0, 1\}$ to denote whether vertex $v$ should be included in the Operating Set (i.e., whether $\mathcal{S}_t = \mathcal{S}_{t-1}$). The value of $I_{v,t}$ is then determined according to Theorem 1.

**Algorithm 1: OperatingSet**$(v, k)$

```
O_t ← O_t ∪ {v}
for each v* ∈ V \ {v} do
    if d(v*, v) ≤ k then
        O_t ← O_t ∪ {v*}
    end
    for each v' ∈ N(v*) do
        if v' ∈ S_{t-1} then
            O_t ← O_t \ {v'}
        end
    end
end
```

### B. Determination of the Operating Set

For every vertex $v \in \mathcal{V}_t$ that is included in the Operating Set $\mathcal{O}_t$, it is straightforward to also include all of its $k$-hop neighbors into $\mathcal{O}_t$. However, in this section we will analytically investigate whether we can further reduce the size of $\mathcal{O}_t$ by excluding some of the neighbors of $v$ from $\mathcal{O}_t$.

Our basic insight of reducing the size of $\mathcal{O}_t$ is that the scheduling result within $\mathcal{O}_t$, when being combined with the previous scheduling results over the remaining components of the network (i.e., $\mathcal{S}_{t-1} \setminus \{\mathcal{S}_{t-1} \cap \mathcal{O}_t\}$, should still be an IS of the conflict graph $G$. Therefore, for any $v \in \mathcal{O}_t$, we should ensure that $N(v) \cap \mathcal{S}_{t-1} = \emptyset$ and hence avoid any conflict between the new and previous scheduling results.

Based on this insight, the algorithm of determining the Operating Set, with respect to a specific vertex $v \in \mathcal{V}_t$ and the scope of rescheduling (controlled by $k$), is described in Algorithm 1. The algorithm first adds all the $k$-hop neighbors of $v$ into $\mathcal{O}_t$. Afterwards, the Operating Set is further pruned according to $\mathcal{S}_{t-1}$. Such process of determining the Operating Set can be illustrated by the example in Figure 5, where the conflict graph contains 8 vertices and $\mathcal{S}_{t-1} = \{2, 5, 8\}$. At

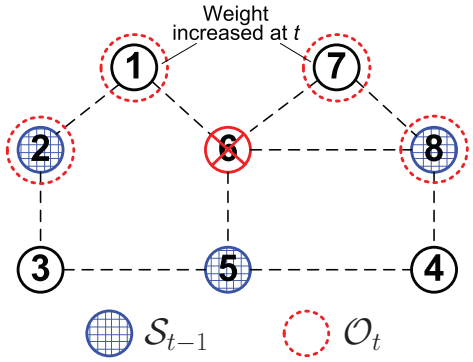



Fig. 5: An example of determining the Operating Set

time slot $t$, the weights of vertex 1 and 7 increase and hence we have $\mathcal{V}_t = \{1, 7\}$. Denote $\mathcal{O}_{i,t}$ as the $\mathcal{O}_t$ when only vertex $i$ changes its weight. Then, by executing Algorithm 1 with $k = 1$, we have $\mathcal{O}_{1,t} = \{1, 2, 6\} \setminus \{6\} = \{1, 2\}$, where vertex 1 changes its weight and vertex 2 and 6 are its neighbors. According to Algorithm 1, vertex 6 is pruned from $\mathcal{O}_t$ because it is a neighbor of vertex 5 which is in $\mathcal{S}_{t-1}$. Otherwise, if vertex 6 is involved in $\mathcal{O}_t$ and selected into $\mathcal{S}_t$, it will conflict with vertex 5 so that $\mathcal{S}_t$ will not be an IS of $G$. Similarly, we have $\mathcal{O}_{7,t} = \{6, 7, 8\} \setminus \{6\} = \{7, 8\}$. Finally, we determine the Operating Set $\mathcal{O}_t$ as $\mathcal{O}_{1,t} \cup \mathcal{O}_{7,t} = \{1, 2, 7, 8\}$.

### C. Scheduling Algorithm

Built on Algorithm 1, our proposed scheduling algorithm with limited operations is shown in Algorithm 2. This algorithm returns the MWIS of the conflict graph $G$ at time slot $t$ by only finding the MWIS within the Operating Set $\mathcal{O}_t$, and we utilize the analytical results derived from the previous sections to minimize the size of $\mathcal{O}_t$. More specifically, for each vertex $v$ in the Variation Set $\mathcal{V}_t$, we first determine whether it needs to be involved in the scheduling process based on Theorem 1, and then generate the Operating Set using Algorithm 1. Afterwards, the MWIS of $\mathcal{O}_t$, named $\mathcal{S}_{in,t}$, is obtained by running existing MWIS algorithms over $\mathcal{O}_t$. $\mathcal{S}_t$ is then constructed by combining $\mathcal{S}_{in,t}$ with $\mathcal{S}_{t-1}$.

In practice, our scheduling algorithm can be operated in a fully distributed manner over mobile nodes in the wireless network. First, a mobile node can easily decide the value of $I_{v,t}$ for each of its wireless links based on its local knowledge about these links and $\mathcal{S}_{t-1}$ in the previous time slot. Second, for each wireless link being rescheduled, a mobile node can efficiently execute Algorithm 1 by exchanging information about other wireless links within its $k$-hop neighborhood. Third, there has been a large body of distributed MWIS algorithms that can provide optimized network throughput with controllable communication overhead [9], [17], [20].

Finally, the correctness of our proposed network scheduling approach is proved by the following theorem.

**Theorem 2.** *The vertex set $\mathcal{S}_t$ obtained by Algorithm 2 is an IS of the conflict graph $G$ at time slot $t$.*

*Proof:* We prove this theorem by finding a contradiction. Since $\mathcal{S}_t$ is the union of two ISs $\mathcal{S}_{in,t}$ and $\mathcal{S}_{out,t}$, if $\mathcal{S}_t$ is not an IS, then there exists a vertex $m \in \mathcal{S}_{in,t}$ and vertex

**Algorithm 2:** Scheduling Algorithm

```
for each v ∈ V_t do
    if I_{v,t} = 1 then
        OperatingSet(v, k)
    end
end
S_{in,t} ← MWIS(O_t)
S_{out,t} ← S_{t-1} \ {S_{t-1} ∩ O_t}
S_t ← S_{in,t} ∪ S_{out,t}
```

$n \in \mathcal{S}_{out,t}$ such that $m$ and $n$ are neighbors on $G$. Since $\mathcal{S}_{in,t} \subset \mathcal{O}_t$, we have $m \in \mathcal{O}_t$. This indicates that there exists a vertex $m \in \mathcal{O}_t$, such that $m$ has at least one neighbor in $\mathcal{S}_{t-1}$, which contracts with the determination process of the Operating Set being described in Algorithm 1. Therefore, $\mathcal{S}_t$ is an IS of $G$ at time slot $t$. ■

## V. Performance Analysis

In this section, we conduct formal analysis on the performance of our proposed network scheduling approach, and show that our scheduling algorithm described in Section IV provides a $\frac{1}{2}$-approximation to the optimal solution at the global network scale with any possible network topology and scope of rescheduling. Without loss of generality, we denote $\mathcal{S}_t$ as the IS being produced by our scheduling algorithm and $\mathcal{S}_t^*$ as the actual MWIS of $G$ at $t$. Then, we denote $\mathcal{S}_{in,t}^* = \mathcal{S}_t^* \cap \mathcal{O}_{in,t}$ and $\mathcal{S}_{out,t}^* = \mathcal{S}_t^* \setminus \mathcal{S}_{in,t}^*$. Note that, although $\mathcal{S}_{in,t}$ produced by existing MWIS algorithm is the actual MWIS of $\mathcal{O}_t$ at time $t$, $\mathcal{S}_t$ may not be the IS of $G$ with the maximum weight at time $t$, when it is produced by combining $\mathcal{S}_{in,t}$ and $\mathcal{S}_{out,t}$.

We start our analysis with the following theorem.

**Theorem 3.** *If a vertex $v$ in $G$ increases its weight at time $t$, i.e., $w_{v,t} > w_{v,t-1}$, we have*

$$W(\mathcal{S}_t)/W(\mathcal{S}_t^*) \geq \frac{1}{2} \tag{1}$$

*if $v \notin \mathcal{S}_{t-1}$.*

*Proof:* As $\frac{W(\mathcal{S}_t)}{W(\mathcal{S}_t^*)} = \frac{W(\mathcal{S}_{in,t})+W(\mathcal{S}_{out,t})}{W(\mathcal{S}_{in,t}^*)+W(\mathcal{S}_{out,t}^*)}$, we prove this theorem in two steps.

First, since $\mathcal{S}_{in,t}$ is obtained by running existing distributed MWIS algorithm within the Operating Set $\mathcal{O}_t$, we have $W(\mathcal{S}_{in,t}) \geq W(\mathcal{S}_{in,t}^*)$. Hence, $W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t}) \geq W(\mathcal{S}_{in,t}^*)$.

Second, we prove that $W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t}) \geq W(\mathcal{S}_{out,t}^*)$ by finding a contradiction. Since $W(\mathcal{S}_{in,t}) \geq W(\mathcal{S}_{t-1} \cap \mathcal{O}_{in,t})$ and $\mathcal{S}_{out,t} = \mathcal{S}_{t-1} \setminus \{\mathcal{S}_{t-1} \cap \mathcal{O}_{in,t}\}$, we have

$$\begin{aligned} W(\mathcal{S}_t) &\geq W(\mathcal{S}_{t-1} \cap \mathcal{O}_{in,t}) + W(\mathcal{S}_{t-1} \setminus \{\mathcal{S}_{t-1} \cap \mathcal{O}_{in,t}\}) \\ &= W(\mathcal{S}_{t-1}). \end{aligned}$$

Now, if $W(\mathcal{S}_t) = W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t}) < W(\mathcal{S}_{out,t}^*)$, then $W(\mathcal{S}_{t-1}) < W(\mathcal{S}_{out,t}^*)$, this contradicts with the fact that $\mathcal{S}_{t-1}$ is the MWIS of the conflict graph at slot $t-1$. Hence, $W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t}) \geq W(\mathcal{S}_{out,t}^*)$.

Therefore, by combining the results of the above two steps, we have

$$\frac{W(\mathcal{S}_t)}{W(\mathcal{S}_t^*)} \geq \frac{W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t})}{2(W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t}))} = \frac{1}{2}.$$

■

Similarly, we also proved that when some vertices in $\mathcal{V}_t$ decrease their weights at $t$, the same approximation ratio holds.

**Lemma 3.** *For a vertex $v$ in $G$ which decreases its weight at time $t$, i.e., $w_{v,t} < w_{v,t-1}$, if $v \in \mathcal{S}_{t-1}$ and $v \notin \mathcal{S}_t$, we have $v \notin \mathcal{S}_t^*$.*

*Proof:* Lemma 3 can be proved by finding a contradiction. If $v \in \mathcal{S}_t^*$, since $v$ is the only vertex that changes its weight at time $t$, we have $\mathcal{S}_t^* = \mathcal{S}_{t-1}$. Otherwise, there must be a vertex set $\mathcal{S}^*$ such that $\mathcal{S}_t^* = \{v\} \cup \mathcal{S}^*$ and $w_{v,t-1} + W(\mathcal{S}^*) > W(\mathcal{S}_{t-1})$, which contradicts with that $\mathcal{S}_{t-1}$ is the MWIS of $G$ at slot $t$.

If $\mathcal{S}_t^* = \mathcal{S}_{t-1}$, we have $\mathcal{S}_{out,t}^* = \mathcal{S}_{out,t-1} = S_{out,t}$ because the previous scheduling result $\mathcal{S}_{out,t-1}$ outside of $\mathcal{O}_t$ is reused to produce $\mathcal{S}_t$. Also, since $\mathcal{S}_t^*$ is the actual MWIS of $G$ at slot $t$, we have $W(\mathcal{S}_t^*) \geq W(\mathcal{S}_t)$ which indicates that

$$W(\mathcal{S}_{in,t}^*) + W(\mathcal{S}_{out,t}^*) > W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t}),$$

we have $W(\mathcal{S}_{in,t}^*) > W(\mathcal{S}_{in,t})$. However, this contradicts with the fact that $\mathcal{S}_{in,t}$ is the MWIS of the Operating Set $\mathcal{O}_t$. Hence we prove that $v \notin \mathcal{S}_t^*$. ■

Based on Lemma 3, we immediately get the following lemma:

**Lemma 4.** *For a vertex $v$ in $G$ which decreases its weight at time $t$, i.e., $w_{v,t} < w_{v,t-1}$, if $v \in \mathcal{S}_{t-1}$ and $v \notin \mathcal{S}_t$, then we have $W(\mathcal{S}_t) > W(\mathcal{S}_{out,t}^*)$.*

*Proof:* From Lemma 3, we have that $v \notin \mathcal{S}_t^*$. We prove $W(\mathcal{S}_t) > W(\mathcal{S}_{out,t}^*)$ by finding a contradiction.

We first prove that $W(\mathcal{S}_{t-1}) \leq w_{v,t-1} + W(\mathcal{S}_t)$. Since $\mathcal{S}_{in,t}$ is the MWIS of $\mathcal{O}_t$ and $v \notin \mathcal{S}_t$, we have

$$W(\mathcal{S}_{in,t}) \geq W(\mathcal{S}_{in,t-1}) - w_{v,t-1}. \tag{2}$$

Meanwhile, since $\mathcal{S}_{out,t} = \mathcal{S}_{out,t-1}$, we have

$$W(\mathcal{S}_{in,t}) + \mathcal{S}_{out,t} \geq W(\mathcal{S}_{in,t-1}) + W(\mathcal{S}_{out,t}) - w_{v,t-1}. \tag{3}$$

Therefore, we derive that $W(\mathcal{S}_{t-1}) \leq w_{v,t-1} + W(\mathcal{S}_t)$ by combining Eq. (2) and Eq. (2).

Then, if $W(\mathcal{S}_t) < W(\mathcal{S}_{out,t}^*)$, we have

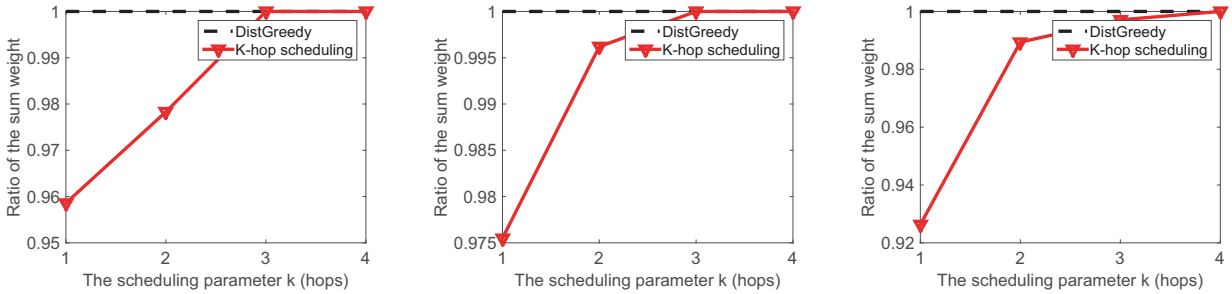
$$\begin{aligned} W(\mathcal{S}_{t-1}) &\leq w_{v,t-1} + W(\mathcal{S}_t) \\ &< w_{v,t-1} + W(\mathcal{S}_{out,t}^*) \end{aligned} \tag{4}$$

which contradicts with that $\mathcal{S}_{t-1}$ is the MWIS of $G$ at time $t-1$. Therefore, the lemma is proved. ■

By combining the results of Lemma 3 and Lemma 4, we have the following theorem:

**Theorem 4.** *When a vertex $v$ in $G$ decreases its weight at time $t$, i.e., $w_{v,t} < w_{v,t-1}$, if $v \in \mathcal{S}_{t-1}$ and $v \notin \mathcal{S}_t$, Eq. (1) still holds.*

*Proof:* First, as $\mathcal{S}_{in,t}$ is the MWIS of $\mathcal{O}_t$, $W(\mathcal{S}_{in,t}) \geq W(\mathcal{S}_{in,t}^*)$. Hence, $W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t}) \geq W(\mathcal{S}_{in,t}^*)$. Second, from Lemma 3 and Lemma 4, we have $W(\mathcal{S}_t) = W(\mathcal{S}_{in,t}) +$




(a) Number of edges = 20 (b) Number of edges = 30 (c) Number of edges = 40

Fig. 6: Average sum of weights in the MWIS over different values of the scheduling parameter $k$

$W(\mathcal{S}_{out,t}) > W(\mathcal{S}^*_{out,t})$. Therefore, by combining the results of the above two steps, we have

$$\frac{W(\mathcal{S}_t)}{W(\mathcal{S}^*_t)} \geq \frac{W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t})}{2(W(\mathcal{S}_{in,t}) + W(\mathcal{S}_{out,t}))} = \frac{1}{2}.$$

■

Note that Theorem 3 and Theorem 4 cover all the possible cases in which network rescheduling is needed. Otherwise, if $v \in \mathcal{S}_{t-1}$ and increases its weight at time $t$, it is trivial to see that $v \in \mathcal{S}_t$ without running the scheduling algorithm. Similarly, $v$ does not need to be rescheduled if it decreases its weight at time $t$ and does not belong to $\mathcal{S}_{t-1}$, because in this case $v$ cannot be in $\mathcal{S}_t$ either. These two theorems can also be recursively applied to every individual vertex in $\mathcal{V}_t$ that changes its weight at time $t$, and hence hold for arbitrary dynamics of wireless networks.

From these two theorems, we can see that our scheduling algorithm, although only being operated over a limited scope of the network, is able to provide guaranteed network throughput. Compared with existing distributed scheduling algorithms (e.g., DistGreedy [9]) which usually provide an approximation ratio of $1/\Delta$ with $\Delta$ denoting the maximum conflict degree of $G$, our algorithm provides a tighter performance bound in most cases. In practice, we can further approximate to optimal throughput by increasing the scope of rescheduling (controlled by the parameter $k$). We will further investigate such tradeoff between network throughput and scheduling overhead via numerical experiments in Section VI.

## VI. Numerical Study

In this section, we evaluate the performance of our scheduling algorithms by conducting simulations over different conflict graphs with large sizes. We measure the performance of the scheduling algorithm using *1)* the average sum of weights in the MWIS and *2)* communication overhead between nodes on the conflict graph.

### A. Simulation Setup

In each experiment, a conflict graph with 30 nodes and a certain number of edges ($M$) is randomly generated, and we control the graph connectivity by varying the value of $M$. We consider the topology of conflict graph as stationary and node weights are valued within the set $\{6, 9, 12, 24, 36, 48, 54\}$. At each time slot, a number of nodes change their weights by randomly picking up another value from the set. In our experiments, each simulation result is obtained from an average of 50 simulation runs, each of which lasts for 50 time slots.

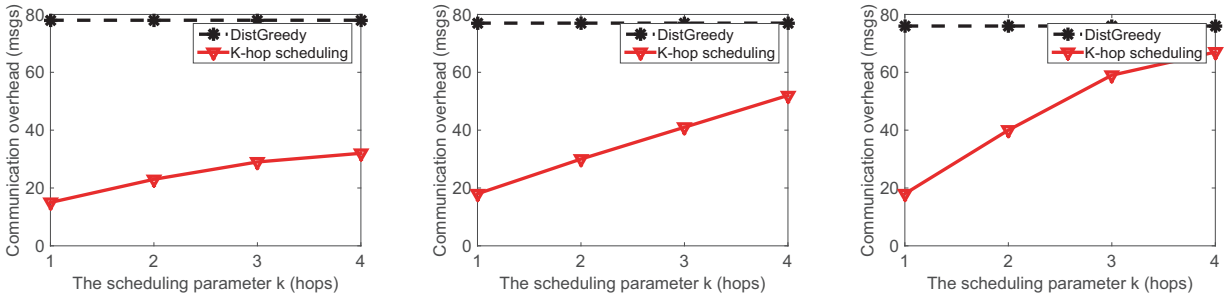
We use DistGreedy [9], which is operated over the entire network at each time slot, as the baseline algorithm for comparison. Similar to our algorithm, DistGreedy can be implemented in a distributed fashion and requires only local information, such as link rates of neighboring links. It also has a time complexity of $O(\log L)$ that grows slowly with the network size. We also use this algorithm to find the MWIS within the Operating Set at each time slot. Then, we compare the performance of our scheduling algorithm with DistGreedy under the following experiment settings.

- **Different values of the scheduling parameter** $k$. As described in Section IV, the parameter $k$ defines the scope of rescheduling. The default value is set to be 4.
- **Different numbers of graph edges**. A large number of edges shows more interference among wireless links. The default number of edges is set to be 30.
- **Different sizes of the Variation Set**. At each time slot, we change the weights of a specific set of nodes. The size of this Variation Set measures the level of network dynamics. The default size of the Variation Set is to 5.

### B. The Effect of Rescheduling Scope

We first evaluate the performance of our scheduling algorithms with different values of the scheduling parameter $k$. We vary the number of edges in the conflict graph between 20 and 40 while keeping the size of the Variation Set to be 5. The sum of weights in the MWIS produced by our algorithms is normalized to that of DistGreedy, and the communication overhead is measured by the number of control messages exchanged between mobile nodes.

The effect of $k$ on the average sum of weights in the MWIS is shown in Figure 6, which shows that the performance of our approach can achieve at least 92% of that of DistGreedy even when $k = 1$. Comparatively when $k = 4$, our approach produces the same level of performance with DistGreedy. When the value of $k$ increases, the performance of our approach improves because a higher value of $k$ leads to a larger Operating Set and a larger scope of rescheduling. In particular, such improvement is also related to the number of edges in the conflict graph. When the number of edges is




(a) Number of edges = 20 (b) Number of edges = 30 (c) Number of edges = 40

Fig. 7: Average communication overhead over different values of the scheduling parameter $k$

small and indicates less interference among wireless links, the performance of our approach approximates closer to that of DistGreedy in all cases.

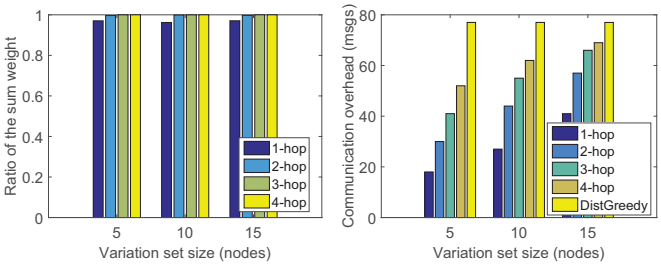
The effect of $k$ on the average communication overhead is shown in Figure 7, which shows that our approach can reduce the communication overhead by up to 75% when $k = 1$ and up to 50% when $k = 4$. Figure 7 also shows that the overhead reduction drops when the value of $k$ increases. The major reason is that a larger value of $k$ incurs a larger Operating Set, in which more control messages will be exchanged between nodes. Furthermore, the overhead of our approach also increases when the number of edges in the conflict graph increases, because the additional network interference requires the scheduling algorithm to generate more control messages for information exchange. Also when the number of edges increases, the same value of $k$ leads to a larger Operating Set and involve more nodes into the rescheduling process.

By combining Figure 6 and Figure 7, we can see that our scheduling algorithm enables flexible tradeoff between the performance and overhead of wireless network scheduling. A larger value of the scheduling parameter $k$ helps improve the sum of weights in the MWIS which directly decides the network throughput, but also incurs a larger number of control messages and increased communication overhead. Therefore, our approach allows the network operator to flexibly adjust the scheduling strategy according to different scenarios and application requirements. In most cases, our approach is able to significantly reduce the scheduling overhead at the cost of a small amount of performance degradation.

Generally speaking, since the variation of throughput degradation of our approach is much smaller than that of communication overhead increases as $k$ increases, the scheduling efficiency, denoted as the ratio of the overall network throughput to the communication overhead, can be viewed as being inversely proportional to the communication overhead, which degrades as $k$ increases. It is noted that the selection of $k$ is determined by the specific application scenarios. For example, single-hop ($k = 1$) network model is well suited to vehicular safety communication in highway [7].

### C. The Effect of the Size of Variation Set

In this section, we evaluate the performance of our approach over different sizes of the Variation Set. The number of edges




(a) Sum of weights in MWIS (b) Communication overhead

Fig. 8: The effect of the size of Variation Set

in the conflict graph is fixed to be 30, and the value of $k$ is set to be 4. The effect of the size of Variation Set on the average sum of weights in MWIS is shown in Figure 8(a), which shows that the size of Variation Set has only little impact on the performance of our approach, no matter what the value of $k$ is. In particular, the performance degradation due to scheduling with limited operations is only noticeable when $k = 1$.

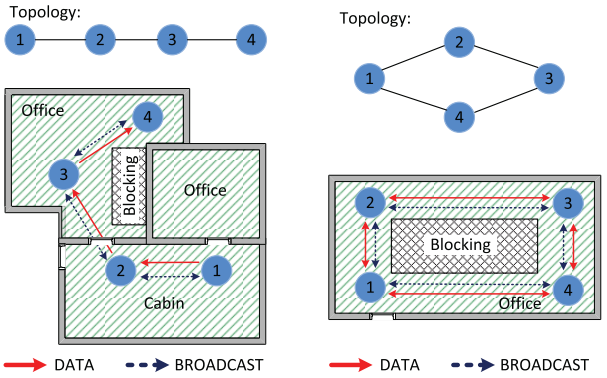
The effect of the size of Variation Set on communication overhead is shown in Figure 8(b), which shows that the communication overhead of our approach increases with a larger size of Variation Set and rescheduling scope. For example, when the size of Variation Set is 5, our approach can reduce 75% of the overhead for $k = 1$, but can only reduce the overhead by 50% if the size of Variation Set grows to 15. However, even when $k = 4$ and the size of Variation Set of 15, the overhead of our approach is still noticeably lower than that of DistGreedy.

## VII. Real-World Experimentation

In this section, we further implement our scheduling algorithms over software-defined radio (SDR) platforms and evaluate our approach over realistic wireless network scenarios.

### A. System Implementation over USRP

We implemented our scheduling algorithms over USRP N210 SDR boards and UBX40 RF daughterboards with the GNURadio toolkit, which realizes an IEEE802.11a WiFi PHY transceiver over the 5.87 GHz frequency band. In addition, we incorporate a complete MAC layer design [6] to implement MAC-layer functionalities such as acknowledgments and carrier sensing. The major challenge of our implementation is that the RF signals harvested by the USRP hardware frontend




(a) Chain topology (b) Square topology

Fig. 9: Two network topologies for experimentation

have to be sent to and processed at the host PC, resulting in long transmission latency and difficulty of meeting the DCF timing requirements. To address this challenge, we follow the same method in [6] and use a scaling factor $\beta(= 5000)$ to emulate the WiFi MAC behaviors.

In order to emulate the realistic behaviors of off-the-shelf wireless devices in practical application scenarios, we incorporate two mainstream rate adaptation algorithms, namely Minstrel [13] and Adaptive Auto Rate Fallback (AARF) [10], into our implementation. Minstrel is a popular WiFi rate adaptation approach that is widely used in commodity WiFi transceivers and has been integrated into the Linux OS kernel. AARF, on the other hand, aims to minimize the packet delivery failure by adaptively adjusting the criteria for adopting higher data rates. In this way, we allow USRP devices to adaptively adjust their link data rates according to the channel conditions, and hence incorporate the practical impact of wireless channel fluctuations on the data rates of wireless links into account.

### B. Experiment Setup

Based on the above implementation, we conduct our experiments with 4 USRP boards which construct two types of wireless network topologies. As shown in Figure 9, these topologies are implemented in separate indoor scenarios with strong channel fading effects. Each network is placed in an office space with a rough size of $5m \times 5m$, and each adjacent pair of USRP boards are placed roughly 4 meters away from each other. We construct these topologies by blocking unwanted wireless links between USRP boards. In order to efficiently realize such blocking, we adaptively adjust the Tx gain of each USRP board, and also utilize indoor obstacles such as walls, metal objects and furniture. Our testing results show that a node can only overhear a small portion (less than 2%) of packets sent from its unwanted neighbors.

In our experiments, we compare the scheduling performance and overhead of our algorithms with DistGreedy [9]. During each experiment that lasts 40 minutes, all the USRP senders keep transmitting WiFi packets of 100 bytes to the corresponding receiver. The network queuing, then, is automatically handled by the MAC layer at USRP. The scheduling performance is represented by the network throughput measured as the number of data frames being successfully received at USRP receivers. The scheduling overhead is measured as the number of broadcast frames, which contain the wireless link information being used for scheduling, sent by USRP senders.

We implement the DistGreedy algorithm to be periodically running at all the 4 nodes in the network, and it enables each sender to periodically broadcast its link information to its neighbors. Upon receiving these broadcast frames, each node will then adjust its backoff window by comparing its link data rate with that of its neighbors, and a USRP sender broadcasts one frame for scheduling as soon as it has sent ten data frames out. For fair comparison, we adopt the same rule for setting up the backoff window in our scheduling algorithm. Since there are only 4 nodes in the network, we set $k = 1$ for our scheduling algorithm, which means that when the status of one wireless link changes, only itself and its neighbor links are involved for rescheduling.

### C. Experiment Results

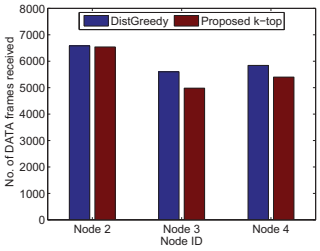
We conduct our experiments with different network topologies and rate adaptation approaches, and the experiment results are shown in Figures 10 to 13. In all the experiments, the network throughput provided by our algorithms approximates to that of DistGreedy, and the degradation of network throughput is less than 10%. Comparatively, by constraining the scope of network scheduling, our algorithms reduces the communication overhead by more than 50% at all USRP senders.

From these results, we conclude that our scheduling algorithms can provide guaranteed network throughput with low overhead in different network topologies. When the network topology changes and a node has been connected by a different number of wireless links, we only observe negligible difference in both network throughput and scheduling overhead.

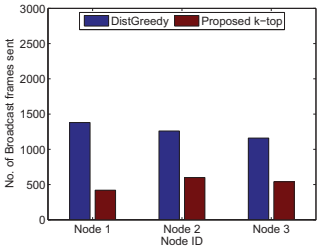
On the other hand, the rate adaptation approach being used in the wireless network has a non-negligible impact on the network throughput and scheduling overhead. Generally speaking, the Minstrel algorithm results in more frequent data rate changes over wireless links in order to keep its statistical table of packet deliveries up to date. Therefore, it leads to slight reduction of network throughput (around 10%) and a noticeable amount of additional broadcast messages produced for rescheduling. Such increase of scheduling overhead also reduces the percentage of overhead reduction achieved by our approach from 50% to 30%.

## VIII. Conclusion

Scheduling in wireless networks is crucial to avoid interference among wireless links and maximize the network throughput. However, existing wireless network scheduling algorithms are frequently executed over the entire network in cases of network dynamics such as channel fading and traffic pattern fluctuations, resulting in a large amount of computation and communication overhead. Instead, in this paper we observe that these network dynamics usually affect only a small portion of the network, and further develop distributed algorithms that schedule wireless networks only within the limited scope where network dynamics occur. Both formal

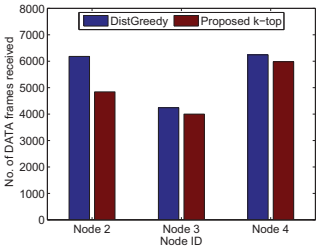



(a) Network throughput

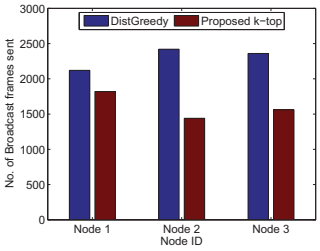



(b) Communication overhead

Fig. 10: Network scheduling with chain topology and AARF rate adaptation

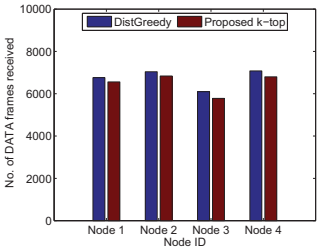



(a) Network throughput

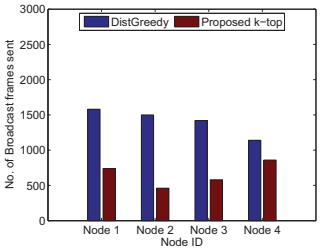



(b) Communication overhead

Fig. 11: Network scheduling with chain topology and Minstrel rate adaptation

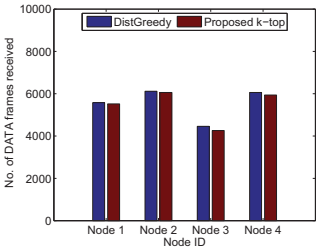



(a) Network throughput

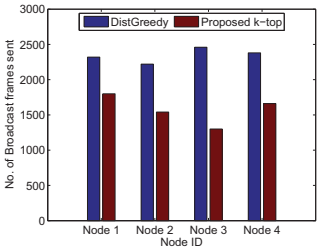



(b) Communication overhead

Fig. 12: Network scheduling with square topology and AARF rate adaptation




(a) Network throughput




(b) Communication overhead

Fig. 13: Network scheduling with square topology and Minstrel rate adaptation

analysis and experimental studies show that our approach is able to significantly reduce the scheduling overhead without impairing the network throughput. Our future work includes expanding the experimentation by involving a larger number of wireless nodes and more types of network topologies.

## REFERENCES


[1] B. Birand, M. Chudnovsky, B. Ries, P. Seymour, G. Zussman, and Y. Zwols. Analyzing the performance of greedy maximal scheduling via local pooling and graph theory. *IEEE/ACM Transactions on Networking*, 20(1):163–176, 2012.

[2] L. X. Bui, S. Sanghavi, and R. Srikant. Distributed link scheduling with constant overhead. *IEEE/ACM Transactions on Networking*, 17(5):1467–1480, 2009.

[3] Q. Chen, Q. Zhang, and Z. Niu. A graph theory based opportunistic link scheduling for wireless ad hoc networks. *IEEE Transactions on Wireless Communications*, 8(10):5075–5085, 2009.

[4] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. In *Proceedings of IEEE INFOCOM*, pages 1794–1803, 2005.

[5] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[6] J. R. Gutierrez-Agullo, B. Coll-Perales, and J. Gozalvez. An ieee 802.11 mac software defined radio implementation for experimental wireless communications and networking research. In *IFIP Wireless Days*, 2010.

[7] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 ghz dsrc-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, 2006.

[8] L. Jiang and J. Walrand. A distributed CSMA algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 18(3):960–972, 2010.

[9] C. Joo, X. Lin, J. Ryu, and N. B. Shroff. Distributed greedy approximation to maximum weighted independent set for scheduling with fading channels. In *Proceedings of ACM MobiHoc*, pages 89–98, 2013.

[10] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 rate adaptation: A practical approach. In *Proceedings of MSWiM*, 2004.

[11] M. Leconte, J. Ni, and R. Srikant. Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks. *IEEE/ACM Transactions on Networking*, 19(3):709–720, 2011.

[12] X. Liu, E. K. Chong, and N. B. Shroff. A framework for opportunistic scheduling in wireless networks. *Computer networks*, 41(4):451–474, 2003.

[13] M. Mirza, P. Barford, X. Zhu, S. Banerjee, and M. Blodgett. Fingerprinting 802.11 rate adaption algorithms. In *Proceedings of IEEE INFOCOM*, pages 1161–1169, 2011.

[14] W. Nam, D. Bai, J. Lee, and I. Kang. Advanced interference management for 5G cellular networks. *IEEE Communications Magazine*, 52(5):52–60, 2014.

[15] J. Ni, B. Tan, and R. Srikant. Q-CSMA: queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks. *IEEE/ACM Transactions on Networking*, 20(3):825–836, 2012.

[16] I. C. Paschalidis, F. Huang, and W. Lai. A message-passing algorithm for wireless network scheduling. *IEEE/ACM Transactions on Networking*, 23(5):1528–1541, 2015.

[17] P. E. Santacruz, V. Aggarwal, and A. Sabharwal. Beyond interference avoidance: Distributed sub-network scheduling in wireless networks with local views. In *Proceedings of IEEE INFOCOM*, pages 460–464, 2013.

[18] G. Sharma, R. R. Mazumdar, and N. B. Shroff. On the complexity of scheduling in wireless networks. In *Proceedings of ACM MobiCom*, pages 227–238, 2006.

[19] Y. Shi, Y. T. Hou, J. Liu, and S. Kompella. How to correctly use the protocol interference model for multi-hop wireless networks. In *Proceedings of ACM MobiHoc*, pages 239–248, 2009.

[20] C. Singh, A. Nedic, and R. Srikant. LP-relaxation based distributed algorithms for scheduling in wireless networks. In *Proceedings of IEEE INFOCOM*, pages 1905–1913, 2014.

[21] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992.

[22] R. Urgaonkar and M. J. Neely. Opportunistic scheduling with reliability guarantees in cognitive radio networks. *IEEE Transactions on Mobile Computing*, 8(6):766–777, 2009.

[23] P.-J. Wan, O. Frieder, X. Jia, F. Yao, X. Xu, and S. Tang. Wireless link scheduling under physical interference model. In *Proceedings of IEEE INFOCOM*, pages 838–845. IEEE, 2011.

[24] P.-J. Wan, X. Jia, G. Dai, H. Du, and O. Frieder. Fast and simple approximation algorithms for maximum weighted independent set of links. In *Proceedings of IEEE INFOCOM*, pages 1653–1661, 2014.

[25] X. Wu, R. Srikant, and J. R. Perkins. Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks. *IEEE Transactions on Mobile Computing*, 6(6):595–605, 2007.

[26] Y. Yi, A. Proutière, and M. Chiang. Complexity in wireless scheduling: Impact and tradeoffs. In *Proceedings of ACM MobiHoc*, 2008.