Joint Scheduling and Multiflow Maximization in Wireless Networks

Yanxiao Liu, Shenghao Yang and Cheuk Ting Li

***Abstract*—Towards the development of 6G mobile, it is promising to integrate a large number of devices from multi-dimensional platforms, and it is crucial to efficiently measure the performance of the network, i.e., solving the maximum multiflow problem in multi-hop networks. The problem is usually solved in two steps: first find the scheduling rate region, and then find the maximum multiflow that can be supported by achievable link rates. However, the scheduling problem is NP-hard, which makes solving the multiflow problem in large-scale networks computationally prohibitive. In this paper, we provide efficient algorithms that can provably output the optimal solutions, without the need of the whole scheduling rate region. Our unified framework works in general multi-source multi-sink networks, with network coding performed on intermediate nodes, and propagation delays can be utilized in scheduling. We prove our algorithms output exact-optimal solutions (instead of approximations) in a finite number of iterations, and perform various experiments to show the advantages over conventional approaches.**

***Index Terms*—multihop network, maximum multiflow problem, network coding, wireless scheduling, network throughput**

I. Introduction

Over the past years, both the number of users in wireless networks and the services to be provided have experienced significant growth. To deploy the next generation mobile system, it is expected that a massive connectivity and emerging applications should be supported. To design large-scale, highly-connected wireless systems and to measure their performance, it is crucial but very difficult to find the maximum multiflow (MMF) and maximum concurrent multiflow (MCMF) that can be supported by collision-free link schedules of the network.

The MMF (or MCMF) problem addresses the maximum total (or concurrent) throughput between multiple source nodes and sink nodes supported by achievable link rates, and is one of the problems at the heart of network communication theory. The key issues affecting the performance of multihop wireless networks are the wireless interference and the scheduling rate region. To solve the MMF (or MCMF) problem, traditional methods usually first use a *link conflict graph* to model the wireless interference [1], then convert the scheduling problem to the maximum independent set problem (which is a well-known NP-hard problem that is also hard to approximate [2]) in the graph for searching collision-free schedules, and finally calculate the maximum (concurrent) throughput under the constraints described by the scheduling rate region. Due to the hardness of scheduling, it is unrealistic to solve the MMF and MCMF problems in large-scale networks by this way. In [3] it has been shown that both problems are NP-hard even in very simple settings. Although there exist joint optimization methods with low complexities, most of them only *approximate* the throughput with certain constraints (possibly in a decentralized manner). However, we are interested in deriving the *exact-optimal* solutions in a finite number of iterations.

In this paper, we design practical algorithms that efficiently solve the MMF and MCMF problem, which *provably* provide optimal solutions (instead of approximations by distributed optimization frameworks). Our algorithms work in general multi-hop networks with network coding [4]–[6] on intermediate nodes, and jointly calculate the maximum (concurrent) multiflow and the scheduling rate region by employing a decomposition method. Compared to the conventional method (first calculate the scheduling rate region, and then solve the MMF and MCMF problems), we only need a subset of the scheduling rate region, and hence can be much more efficient in practice. Except theoretic analysis, we use experiments to show our advantages over the two-step method. Our unified framework can be easily generalized to the networks with non-negligible propagation delays (e.g., underwater and deep-space), for which it has been recently discussed [7]–[11] that scheduling region can be significantly improved but becomes even harder to compute, and all the advantages (generality, efficiency and optimality) of our approach are maintained.

We briefly summarize our contributions and advantages against existing results.

- We provide practical algorithms for the MMF and MCMF problems, which can be much more efficient than [1], [3].
- We solve the *multiple multicast* problem with network coding [4]–[6] allowed, which is the most general case in multi-source multi-sink networks. In comparison, existing works [1], [3], [12] only study multiple unicast case.
- We prove our algorithms output exact-optimal solutions, instead of giving approximations or converging to optimum by decentralized optimization methods [13]–[16].
- Our unified framework covers the case where large propagation delays are utilized in scheduling [7], [8], [10], [11], which is promising in certain (e.g., underwater) networks.

The remainder of the paper is organized as follows. We first provide a comprehensive literature review in Section II. In Section III, we describe the network model and problem formulation. We propose our algorithms in Section IV and prove the optimality. The performance evaluation is in Section V.

II. Related Works

The literature review includes three parts: First, we discuss the MMF and MCMF problems. Second, we review related optimization frameworks and network coding. Finally, we

review recent works on utilizing non-negligible propagation delays in certain (e.g., underwater and deep-space) networks.

A. Maximum Multiflow Problem

The maximum multiflow (MMF) and the maximum concurrent multiflow (MCMF) are core problems in the theory of network communications. The MMF problem studies the maximum throughput between selected source nodes and sink nodes [1], [3], and the maximum concurrent multiflow problem [17] models the case where every sender-receivers session transmits messages concurrently. The NP-hardness of both problems have been proved in [3], even in very simple network settings. In [18], [19], both the MMF and MCMF problems are discussed under the interference model that nodes cannot transmit and receive simultaneously. By enforcing interference constraints on links, [20] guarantees the schedulability and develops constant-approximation algorithms. More linear programming formulation and approximation algorithms can be found in [3], [17], [18]. In [21], the MMF and MCMF problems are discussed by dividing the cases to full-duplex systems and half-duplex systems, both of which are covered by our interference model in this paper. The MMF problem has been extended to unicast networks with network coding [12], where the network coding [4]–[6] is treated as a scheme to decrease the impact of wireless interference.

To support the (concurrent) multiflow in networks, it is usually required to first find achievable link rates, which forms the scheduling rate region. To solve the scheduling problem, in [1], the effects of wireless interference can be modeled by a *conflict graph*, and the scheduling problem is equivalent to searching all the maximal independent sets in the conflict graph, which is an NP-hard problem that is also hard to approximate [2]. Due to this hardness, it is computationally prohibitive to calculate the scheduling region before solving the MMF (or MCMF) problem. In practice, optimization algorithms (possibly in a decentralized manner) can be used to approximate the solutions with a low complexity (see Section II-B as follows), but our objective is to find the exact maximum (concurrent) multiflow value in an efficient way.

B. Joint Optimization Frameworks and Network Coding

Existing works on the MMF (or MCMF) problem [1], [3], [12] only study the multiple unicast case, i.e., each source node is paired with one sink node. However, we consider multiple multicast in general multi-source multi-sink networks, where each of a number of source nodes transmits a message to a set of sink nodes. In this scenario, network coding [4]–[6] is an effective technique to improve the network performance, and the throughput can increase up to several folds [22], [23]. The joint consideration of throughput, scheduling and network coding has been widely studied in [13]–[16], [24] for various objectives, e.g., maximizing throughput or minimizing the energy consumption under certain constraints. These approaches are either converging-to-optimal with respect to some constraints or only approximate the solutions.

In [24], the authors decompose the joint optimization of scheduling and network coding into two subproblems, similar with us decomposing the joint MMF (and MCMF) and scheduling problems. However, some differences are as follows. Except we provide a unified framework that also covers the case where the propagation delays are non-negligible and utilized in scheduling (see Section II-C as follows), we study the multi-source (instead of single source) multi-sink case, where the trade-off between the rates of the sources becomes an important factor of consideration. Moreover, the algorithm in [24] is an iterative algorithm that only converges to the optimum, but our algorithms provably output the exact optimum in a finite number of iterations (also see Remark 3).

C. Networks with Non-negligible Propagation Delays

In the existing theory of terrestrial wireless communications, though the communication media (e.g., radio, light, sound) have nonzero propagation delays, they are usually short and treated as a factor of interference [25]. However, in certain environments, e.g., deep-space and underwater networks, the propagation delays can be significantly longer. For example, for sound (whose speed is about 1.5 kilometers per second) to propagate over a distance of 3 kilometers, the delay can be about 2 seconds. Recent studies [7]–[11], [26], [27] show that it is promising to *utilize* the delays to significantly improve the network performance. For such scenarios, mixed integer linear programming for some heuristic algorithms [11], [27] and dynamic-programming based algorithms [10] were proposed.

In [7], [8], by extending the conflict graph [1] to a *weighted* graph, the scheduling rates can be exactly characterized with even higher complexity. We utilize their formulation in our framework, and provide an efficient algorithm that solves the MMF and MCMF problems while also utilizing the delays to improve the scheduling rates. Existing methods for the MMF and MCMF problems cannot be directly extended to this case.

III. Network Model

In this section, we define the network model and formulate the MMF and MCMF problems.

A. Network Model

We use a link-wise network model [1], [3], [7], [8], where each link is associated with a *collision set*, including all the links that can be interfered by it. It is called the binary interference model [1], [7], and it is not difficult to cover the physical interference model by signal-to-interference-and-noise ratio [8], [15]. We assume the network is acyclic and *discrete* [7], [8], [26] in the sense that time is slotted and the link delays are multiples of a length of a time slot, which is justified in [26]. The intermediate nodes can wait until enough packets are collected before performing coding on the packet.

The network can be modeled by a tuple $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{I}, D)$, where $\mathcal{V}$ is the *node set*, $\mathcal{L} \subseteq \mathcal{V}^2$ is the *link set*, $\mathcal{I} = (\mathcal{I}(l), l \in \mathcal{L})$ is the set of *collision sets* where $l' \in \mathcal{I}(l)$ if l' is in the interference range of l and $D : \mathcal{L}^2 \to \mathbb{Z}$ is the link-wise delay matrix specifies the delays between links. We assume each link

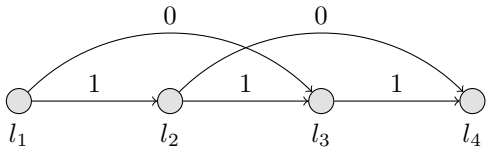

Fig. 1. $\mathcal{N}_{4,1}^{D=1}$: nodes represents network link, edges represent the collision relations between nodes, and edge weights represent propagation delays.

has a unit bandwidth and allow parallel links between nodes. $(\mathcal{L}, \mathcal{I}, D)$ can form a weighted, directed graph $\mathcal{N}$ where $\mathcal{L}$ is the finite vertex set, (l, l') is an edge if $l' \in \mathcal{I}(l)$, and $D(l, l')$ is the weight on the directed edge (l, l'), which degrades to an unweighted graph $(\mathcal{L}, \mathcal{I})$ if delays are ignored [3], [12]. This graphical approach helps the discussion on our algorithms.

We now describe the communication task over the network $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{I}, D)$. For $k \in \mathbb{N}_+$, let $\mathcal{S} = \{s_1, \ldots, s_k\} \subseteq \mathcal{V}$ be the set of source nodes. We assume the information sources at different source nodes are mutually independent. Each source node s_i is associated with a set of sink nodes $\mathcal{D}_{s_i} \subseteq \mathcal{V}$ that have to decode the information at s_i. We allow a node to be both a source node and a sink node corresponding to another source node. For $i \neq j$, we may have $\mathcal{D}_{s_i} \cap \mathcal{D}_{s_j} \neq \emptyset$, i.e., different source nodes can share same sink nodes. Each link $l \in \mathcal{L}$ represents a point-to-point channel with unit capacity. The sets of input channels and output channels of a node $v \in \mathcal{V}$ are denoted by $\text{In}(v) \subseteq \mathcal{L}$ and $\text{Out}(v) \subseteq \mathcal{L}$, respectively.

To explain the model, we use a line network [7]–[10].

Example 1. Consider an L-hop unicast line network: there are $L+1$ nodes $\mathcal{V} = \{1, 2, \ldots, L+1\}$, with link set

$$\mathcal{L} = \{l_i \triangleq (i, i+1), i = 1, \ldots, L\}.$$

Each link has a unit delay. We use a K-hop interference model: the reception of a node has possible collisions from nodes in K hops. The nodes are half-duplex. The collision set of l_i is

$$\mathcal{I}_K(l_i) = \{l_j : j \neq i, |i+1-j| \leq K\}. \tag{1}$$

The link l_i is active in time slot t if node i sends a signal in time slot t to node $i+1$. Hence the link-wise delay matrix is

$$D(l_i, l_j) = 1 - |i+1-j|. \tag{2}$$

We denote it by $\mathcal{N}_{L,K}^{D=1}$, and it can be represented by a graph, where $\mathcal{N}_{4,1}^{D=1}$ is shown in Figure 1 as an example.

B. Collision-free Schedules and Rate Region

We define collision-free schedules, similar to [7], [8]. Since we assume the time is slotted, when link l is active at time slot t and link $l' \in \mathcal{I}(l)$ is active at time slot $t + D(l, l')$, we say a *collision occurs*. In each time slot we use a binary number to indicate whether a link sends messages or not. Hence we use an infinite binary matrix $S : \mathcal{L} \times \mathbb{N} \to \{0, 1\}$ with rows indexed by $\mathcal{L}$ and columns indexed by $\mathbb{N}$ to specify a *schedule*: $S(l, t) = 1$ indicates that link l is active in time slot t, and $S(l, t) = 0$ indicates it is inactive. $S(l, t)$ has a collision in $\mathcal{N}$ if $S(l, t) = S(l', t + D(l, l')) = 1$ for a certain $l' \in \mathcal{I}(l)$. Otherwise $S(l, t)$ is *collision free*. A schedule S is collision free if $S(l, t)$ is collision free for all (l, t). There are different (though with similar ideas) definitions if delays are simply ignored, e.g., [3], but we aim to provide a framework general enough to cover the networks with non-negligible delays. For a collision-free schedule S and a link l, the link rate is

$$R_S(l) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} S(l, t). \tag{3}$$

If $R_S(l)$ exists for all $l \in \mathcal{L}$, we call $R_S = (R_S(l), l \in \mathcal{L})$ the *rate vector* of S. For a network $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{I}, D)$, a rate vector $R = (R(l), l \in \mathcal{L})$ is said to be *achievable* if for all $\epsilon > 0$, there exists a collision-free schedule S such that $R_S(l) > R(l) - \epsilon$ for all $l \in \mathcal{L}$. For a link l, the rate $R(l)$ can stand for the maximum number of information symbols that can be sent on the channel per time slot. Then each achievable rate vector can be viewed as a rate constraint for the network.

The collection $\mathcal{R}(\mathcal{N})$ of all the achievable rate vectors is called the *(scheduling) rate region* of $\mathcal{N}$. We may use $\mathcal{R}$ instead of $\mathcal{R}(\mathcal{N})$ to simplify the notation when the context is clear. It is proved in [7] that $\mathcal{R}$ is a convex polytope, and can be achieved by using periodic collision-free schedules.

C. Problem Formulation

We define the maximum multiflow (MMF) and the maximum concurrent multiflow (MCMF) problems. Most existing literature consider *multiple unicast*, i.e., each source only has one corresponding sink with a certain demand [3], [12], [21]. However, we discuss general *multiple multicast*, with network coding [4]–[6] on intermediate nodes, where the nodes can encode their received data before passing them on.

For each multicast session, one source corresponds to multiple sinks. Though we use network coding to attain the maximum information flow in a session, we do not consider coding *between* sessions (i.e., inter-session network coding [28], [29]) in this paper for the sake of simplicity, since it is in general a hard problem [30] and can even be undecidable [31]. We say

$$F = (F(l) \in \mathbb{N}_{\geq 0} : l \in \mathcal{L}) \tag{4}$$

is a valid flow from source node $s \in \mathcal{V}$ to sink node $t \in \mathcal{V}$ with respect to a rate vector R if it satisfies:

- $0 \leq F(l) \leq R(l)$ for all $l \in \mathcal{L}$, i.e., the flow along link l does not exceed the rate constraint $R(l)$.
- The flow conservation equation $\sum_{l \in \text{In}(v)} F(l) = \sum_{l \in \text{Out}(v)} F(l)$ for all $v \in \mathcal{V} \backslash \{s, t\}$.

We see the flow $\sum_{l \in \text{Out}(s)} F(l)$ out of s equals to the flow $\sum_{l \in \text{In}(t)} F(l)$ into t, and this value is called the *value* of F, denoted as $\text{val}(F)$. We say F is a *max-flow* from s to t with respect to R if F is a flow and has a value no smaller than the value of any other flow from s to t with respect to R.

1) Maximum Multiflow (MMF) Problem: Consider a source s_i multicasts a message to nodes in the set $\mathcal{D}_{s_i} = \{t_{i,1}, \ldots, t_{i,k_i}\}$, with network coding [4]–[6], at a rate

$$v_i = \min_j \text{val}(F_{i,j}),$$

where $F_{i,j}$ is a flow from s_i to $t_{i,j}$ for each j and the rate of communication along link l is $\max_j F_{i,j}(l)$.

We now put the flows from the source nodes $s_1, \ldots, s_k$ together. Fix a rate vector R. Link l has to accommodate all these k flow requirements simultaneously, i.e., $\sum_{i=1}^{k} \max_j F_{i,j}(l) \leq R(l)$ for all $l \in \mathcal{L}$. We maximize the sum of the rates of multicasting these k sources, and it is called the maximum multiflow (MMF) problem, which is formulated by the following linear program, combining the linear program for multiple unicast [1], [3], [12] and the program for single multicast [24]:

LP-MMF($\tilde{\mathcal{R}}$) :

$$\begin{aligned}
&\text{maximize} \quad \sum_{i=1}^{k} v_i && (5)\\
&\text{subject to}\\
&F_{i,j} \text{ is a valid flow, } \forall i \in [k], j \in [k_i],\\
&\sum_{l \in \mathrm{Out}(s_i)} F_{i,j}(l) = \sum_{l \in \mathrm{In}(t_{i,j})} F_{i,j}(l) = v_i, \ \forall i \in [k], j \in [k_i],\\
&G_i(l) \geq F_{i,j}(l), \ \forall l \in \mathcal{L}, i \in [k], j \in [k_i],\\
&\sum_{i=1}^{k} G_i(l) \leq R(l), \ \forall l \in \mathcal{L}, && (6)\\
&R \in \tilde{\mathcal{R}},
\end{aligned}$$

where $[k] := \{1, \ldots, k\}$, $k, k_i \in \mathbb{N}_+, \forall i$. The linear program takes a polytope $\tilde{\mathcal{R}}$ (a subset of the scheduling rate region) as an input. The variables $G_i(l), i \in [k], l \in \mathcal{L}$ are used to impose the constraint $\sum_{i=1}^{k} \max_j F_{i,j}(l) \leq R(l)$. The constraint (6) gives a dual vector that will be used in our algorithms, and the dual variable corresponding to link l means how sensitive the optimization objective is to the rate constraint $R(l)$.

2) Maximum Concurrent Multiflow (MCMF) Problem: While the MMF problem is to find the link schedule that can support the maximum total rate of transmission of the sources, the maximum concurrent multiflow (MCMF) problem is to find the link schedule such that all the sources can transmit concurrently at the maximum rate [3], [17], [18]. The settings of MCMF problems in [3], [18] are also for multiple unicast.

More generally, instead of maximizing the sum rate $\sum_{i=1}^{k} v_i$, we maximize ϕ such that the source s_i can multicast at a rate $v_i = \phi\gamma_i$, where γ_i is the desired traffic rate at s_i. The MCMF problem is formulated as follows [3], [24]:

LP-MCMF($\tilde{\mathcal{R}}$) :

$$\begin{aligned}
&\text{maximize} \quad \phi && (7)\\
&\text{subject to}\\
&F_{i,j} \text{ is a valid flow, } \forall i \in [k], j \in [k_i],\\
&\sum_{l \in \mathrm{Out}(s_i)} F_{i,j}(l) = \sum_{l \in \mathrm{In}(t_{i,j})} F_{i,j}(l) = \phi\gamma_i, \ \forall i \in [k], j \in [k_i],\\
&G_i(l) \geq F_{i,j}(l), \ \forall l \in \mathcal{L}, i \in [k], j \in [k_i],\\
&\sum_{i=1}^{k} G_i(l) \leq R(l), \ \forall l \in \mathcal{L}, && (8)\\
&R \in \tilde{\mathcal{R}}.
\end{aligned}$$

We also utilize the dual vector given by (8) in the algorithms.

IV. Algorithms

In this section, we provide two algorithms for the networks with and without propagation delays, in a unified framework. We prove both algorithms output exact-optimal solutions (instead of approximations) in a finite number of iterations.

A. Algorithm for Networks with Negligible Delays

We first discuss the conventional case where the propagation delays are negligible. Our algorithm calculates the MMF or MCMF without the need of the whole scheduling rate region, and provably outputs the exact-optimal MMF or MCMF.

Since a collision-free schedule can be found by searching an independent set in the graph $(\mathcal{L}, \mathcal{I})$ [7], we attach a weight $a_i \geq 0$ to link l_i, and would like to maximize the weighted total rate, i.e., for the scheduling rate region $\mathcal{R}$, we solves

$$\arg\max_{R \in \mathcal{R}} \langle \mathbf{a}, R \rangle, \tag{9}$$

where $\mathbf{a} = (a_i)_{i=1,\ldots,|\mathcal{L}|}$, and $\langle \cdot, \cdot \rangle$ is the inner product.

Remark 1. For (9), the objective is to maximize the weighted sum rate instead of just the sum rate, since a different weight vector could be used in each iteration of our algorithms (see step 4 of Algorithm 1 or step 5 of Algorithm 2). These weights are also crucial to the graphical approach for Algorithm 2.

This corresponds to a weighted maximal independent set problem [3], [24] that can be solved by integer linear programming (ILP) by maximizing over $S(l_i) \in \{0, 1\}$ for $l_i \in \mathcal{L}$:

$$\begin{aligned}
\text{ILP:} \quad &\text{maximize} \quad \sum_{i=1}^{|\mathcal{L}|} a_i S(l_i)\\
&\text{subject to} \quad S(l_i) + S(l_j) \leq 1, \quad \forall\, l_i, l_j : l_j \in \mathcal{I}(l_i).
\end{aligned}$$

The solution gives us a maximal independent set of $(\mathcal{L}, \mathcal{I})$, and the corresponding achievable rate vector is $S = (S(l_i), l_i \in \mathcal{L})$, which is a vertex of the scheduling rate region $\mathcal{R}$.

Based on (9), we iteratively search the MMF or MCMF and the scheduling region. Even though our target is the optimal value instead of approximated or converging-to-optimal values, it is unnecessary to find the entire scheduling region before solving the MMF or MCMF problem. Suppose $\mathrm{flow}(\cdot)$ is the function for calculating the MMF or MCMF in a given polytope, which can be a subset of the scheduling region. From $i = 1$, in each iteration, the algorithm works as follows:

1) We start with a subset of the scheduling region $\mathcal{R}_i$, which is formed by the vertices of $\mathcal{R}$ we have known (it is reasonable to assume some rate vectors are known, e.g., by activating the first link all the time and inactivating others, the vector $[1, 0, \ldots, 0]^\intercal$ is achievable). In the first iteration, we start with an arbitrarily chosen rate vector R_1, i.e, $\mathcal{R}_1 = \{R_1\}$. We run the linear program LP-MMF($\mathcal{R}_i$) (or LP-MCMF($\mathcal{R}_i$)) to obtain the dual vector μ_i, corresponding to the constraint in (6) (or (8)).
2) By the ILP, we find a new rate vector R_{i+1} by

$$R_{i+1} = \arg\max_{R \in \mathcal{R}} \langle \mu_i, R \rangle. \tag{10}$$

Algorithm 1 Algorithm for Networks without Delays

Input: a network $(\mathcal{V}, \mathcal{L}, \mathcal{I})$

Output: maximum multiflow v

1: Start with any rate vector $R_1 \in \mathcal{R}$, $\mathcal{R}_1 \leftarrow \{R_1\}$
2: **for** $i = 1, 2, \ldots$ **do**
3: Run $v_i \leftarrow$LP-MMF($\mathcal{R}_i$) (or $v_i \leftarrow$LP-MCMF($\mathcal{R}_i$)) and obtain the dual vector μ_i
4: Run ILP to find $R_{i+1} \leftarrow \arg\max_{R \in \mathcal{R}} \langle \mu_i, R \rangle$
5: $\mathcal{R}_{i+1} \leftarrow \text{conv}\left(\mathcal{R}_i \cup \{R_{i+1}\}\right)$
6: **if** $\langle \mu_i, R_{i+1} \rangle = \max_{R \in \mathcal{R}_i} \langle \mu_i, R \rangle$ **then**
7: **return** v_i

3) We update the subset of the scheduling region by computing the convex hull $\mathcal{R}_{i+1} = \text{conv}\left(\mathcal{R}_i \cup \{R_{i+1}\}\right)$.
4) If $\langle \mu_i, R_{i+1} \rangle = \max_{R \in \mathcal{R}_i} \langle \mu_i, R \rangle$, the algorithm terminates and outputs the last optimal value of LP-MMF($\mathcal{R}_i$) (or LP-MCMF($\mathcal{R}_i$)); otherwise it comes back to step 1 and continues.

We then prove that Algorithm 1 will terminate and output the maximum (concurrent) multiflow in finite iterations.

Theorem 1 (Optimality)**.** *For a network $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{I})$, Algorithm 1 will terminate and output the maximum multiflow (or the maximum concurrent multiflow).*

Proof. The scheduling rate region is a polytope [1] with a finite number of vertices, and hence Algorithm 1 will eventually terminate, since the worst case is that all the vertices are found to solve the maximum (concurrent) multiflow problem.

We then show that the output is optimal; that is, when the algorithm terminates, the output is exactly the MMF or MCMF. We use $\text{flow}(R)$ to denote the function that outputs the MMF or MCMF with respect to a rate vector R.

Denote $\mathcal{R}$ as the entire rate region (which may not need to be found), and $\mathcal{R}_i$ is the subset of $\mathcal{R}$ found until iteration i. Suppose the algorithm terminates at iteration i', i.e., $\langle \mu_{i'}, R_{i'+1} \rangle = \max_{R \in \mathcal{R}_{i'}} \langle \mu_{i'}, R \rangle$. By substituting (10),

$$\max_{R \in \mathcal{R}} \langle \mu_{i'}, R \rangle = \max_{R \in \mathcal{R}_{i'}} \langle \mu_{i'}, R \rangle. \tag{11}$$

Suppose the optimal R in LP-MMF($\mathcal{R}_{i'}$) (or LP-MCMF($\mathcal{R}_{i'}$)) is R^*. Note we can write LP-MMF($\mathcal{R}_{i'}$) as

$$\begin{aligned} &\text{maximize} \;\; \text{flow}(R) - \chi_{\mathcal{R}_{i'}}(\bar{R}) \\ &\text{subject to} \;\; R = \bar{R}, \end{aligned} \tag{12}$$

where $\chi_{\mathcal{R}_{i'}}(\bar{R})$ is the characteristic function (which is 0 if $\bar{R} \in \mathcal{R}_{i'}$, or ∞ otherwise), which forces $\bar{R} \in \mathcal{R}_{i'}$. The dual vector $\mu_{i'}$ corresponding to the constraint in (6) (or (8)) is the same as the dual vector corresponding to the constraint $R = \bar{R}$ in (12). Considering the Lagrangian of (12), at the optimum $(R^*, \bar{R}^*)$, the subgradient satisfies $0 \in \partial_{R^*} \text{flow}(R^*) - \partial_{R^*} \chi_{\mathcal{R}_{i'}}(\bar{R}^*) - \mu_{i'}$ and $0 \in \partial_{\bar{R}^*} \text{flow}(R^*) - \partial_{\bar{R}^*} \chi_{\mathcal{R}_{i'}}(\bar{R}^*) + \mu_{i'}$, and we have $R^* = \bar{R}^*$. Hence, R^* maximizes $\text{flow}(R) - \langle \mu_{i'}, R \rangle$ for $R \in \mathbb{R}_{\geq 0}^{|\mathcal{L}|}$, and maximizes $\langle \mu_{i'}, R \rangle$ for $R \in \mathcal{R}_{i'}$.

Fix any $R' \in \mathcal{R}$. Since R^* maximizes $\text{flow}(R) - \langle \mu_{i'}, R \rangle$ for $R \in \mathbb{R}_{\geq 0}^{|\mathcal{L}|}$,

$$\begin{aligned} \text{flow}(R^*) - \langle \mu_{i'}, R^* \rangle &\geq \text{flow}(R') - \langle \mu_{i'}, R' \rangle \\ &\geq \text{flow}(R') - \langle \mu_{i'}, R^* \rangle, \end{aligned}$$

where the last inequality is by (11) and the fact that R^* maximizes $\langle \mu_{i'}, R \rangle$ for $R \in \mathcal{R}_{i'}$. Therefore, R^* maximizes $\text{flow}(R)$ for $R \in \mathcal{R}$ and the proof is finished. □

Remark 2. Algorithm 1 requires integer linear programming, hence it does not have a polynomial time complexity, which is expected since this problem is NP-hard [1], [3]. Algorithm 1 and Theorem 1 pave the way to the Algorithm 2 (and its optimality) for networks with non-negligible propagation delays.

Remark 3. In [24], an algorithm based on subgradient optimization that decomposes the problem into two parts has been discussed. Though it shares some similarities with ours, our algorithm is guaranteed to find the optimum exactly in a finite number of steps (assuming access to an integer linear programming algorithm), whereas [24] is an iterative algorithm that only converges to the optimum. As we will see in the next section, terminating in a small number of steps is especially important for networks with non-negligible delays, since the update of the subset $\mathcal{R}_i$ of the scheduling region is the bottleneck of the algorithm with exponential time complexity, and should be performed as little as possible.

Example 2. Consider a 2-hop line network $\mathcal{N}_{2,1}^{D=0}$ under the 1-hop interference model. From $R_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^\intercal$, we solve

$$\begin{aligned} \max \quad & v = R(l_1) = R(l_2) \\ \text{s.t. } & 0 \leq R(l_1) \leq 1, \\ & 0 \leq R(l_2) \leq 0, \end{aligned}$$

which gives us a maximum flow of value $v = 0$ in $\mathcal{R}_i = \{R_1\}$. We then use the ILP to find another rate vector:

$$\begin{aligned} \max \quad & R(l_1) + R(l_2) \\ \text{s.t.} \quad & R(l_1) + R(l_2) \leq 1, \end{aligned}$$

which gives $R_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^\intercal$ and $\mathcal{R}_2 = \text{conv}(R_1, R_2)$. Next,

$$\begin{aligned} \max \quad & v = R(l_1) = R(l_2) \\ \text{s.t. } & R(l_1) + R(l_2) \leq 1, R(l_1), R(l_2) \geq 0 \end{aligned}$$

gives us the maximum flow in $\mathcal{R}_2$, $v = 1/2$. We can verify that the resulting dual vector μ_2 in this iteration gives

$$f(\mathcal{R}_2, \mu_2) = \arg\max_{R \in \mathcal{R}_2} f(\mathcal{R}_2, \mu_2),$$

which meets the condition that the algorithm terminates. It is also easy to verify that $1/2$ is indeed the maximum flow value.

B. Algorithm for Networks with Non-negligible Delays

We extend the discussions to networks with non-negligible delays, which is practical in certain (e.g., underwater or deep-space) cases. It relies on a natural extension from the previous framework by utilizing a graphical characterization [7], [8].

As shown in [7]–[11], utilizing the delays can significantly improve the network performance. The key to solve the MMF or MCMF problems in such networks is, we need a function similar to (9) that can output a vertex of the scheduling region in a time complexity at most exponential in $|\mathcal{L}|$ (which in turn will be polynomial in the size of the *scheduling graphs* below), which is more efficient than the cycle-enumeration approach [7] with complexity doubly exponential in $|\mathcal{L}|$.

We review the *scheduling graph* in [7]: For a collision-free schedule matrix S and integers $T \in \mathbb{N}_+$, $k \in \mathbb{Z}$, $S[T,k]$ denotes the submatrix of S consisting of columns $kT, kT+1, \ldots, (k+1)T-1$. If a submatrix S' is formed by T columns of S, its columns are indexed by $0, 1, \ldots, T-1$.

Definition 1 (Scheduling Graph [7])**.** Given a network $\mathcal{N}$ and an integer $T > 0$, the *scheduling graph* $(\mathcal{M}_T, \mathcal{E}_T)$ is a directed graph that is defined as follows: the vertex set $\mathcal{M}_T$ includes all the $|\mathcal{L}| \times T$ binary matrices A such that $A = S[T,0]$ for a certain collision-free schedule S. The edge set $\mathcal{E}_T$ includes all the vertex pairs (A, B) such that $A = S[T,0]$ and $B = S[T,1]$ for a certain collision-free schedule S.

Remark 4. For a network with negligible propagation delays, the scheduling graph $(\mathcal{M}_T, \mathcal{E}_T)$ is a complete graph.

In [7], it has been shown that by choosing $T \geq \max_{l\in\mathcal{L}} \max_{l'\in\mathcal{I}(l)} |D(l,l')|$, calculating the scheduling region is equivalent to searching all the simple cycles in the scheduling graph, which is NP-hard. The scheduling problem then may then even have doubly exponential complexity, since the number of vertices in $(\mathcal{M}_T, \mathcal{E}_T)$ increases exponentially with respect to $|\mathcal{L}|$, and the cycle enumeration in $(\mathcal{M}_T, \mathcal{E}_T)$ is also NP-hard in general.

Instead of enumerating cycles for the scheduling region, we search *maximum-mean-cycles* (which can be solved in a time complexity *polynomial* in the graph size) in a new graph, to find the vertices of the scheduling region. We may only need a few rate vectors to solve the MMF or MCMF problem.

Before describing our approach, we need some graphical concepts. In a directed graph $\mathcal{G}$, a *path* is a sequence of vertices $v_0, v_1 \ldots, v_m$ where for $i = 0, 1, \ldots, m-1$, (v_i, v_{i+1}) is a directed edge. A path is *closed* if $v_0 = v_m$. A *cycle* in $\mathcal{G}$ is a closed path $(v_0, v_1 \ldots, v_m)$ such that $m \geq 1$, $v_i \neq v_j$ for any $0 \leq i \neq j \leq m-1$ and $v_0 = v_m$, i.e., in such a sequence, the only repeated vertices are the first and the last vertices. Note a closed path can be decomposed into a sequence of cycles [32], and this has been used in proving that it suffices to enumerate all the simple cycles for calculating the scheduling rate region [7]. In a graph where each edge is associated with a weight, we say the weight of a directed cycle is the total weight on the edges in the cycle. Then we say the average weight of a directed cycle is the total weight divided by the number of edges in the cycle. The *maximum-mean-cycle* is the cycle in the given weighted, directed graph with the maximum average weight over all directed cycles in the given graph.

It has been proved in [7], [8] that a collision-free, periodic schedule is equivalent to a closed path (which can be decomposed to multiple simple cycles) in $(\mathcal{M}_T, \mathcal{E}_T)$ and vice versa, i.e., the concatenation of a sequence of vertices in $(\mathcal{M}_T, \mathcal{E}_T)$ (which are matrices of size $|\mathcal{L}| \times T$) forms a periodic, collision-free schedule. In this paper, we define a *weighted scheduling graph* and use the maximum-mean-cycle in it to solve (9).

Definition 2 (Weighted Scheduling Graph)**.** Given a weight vector $\mathbf{a} \in \mathbb{R}^{|\mathcal{L}|}$ and a scheduling graph $(\mathcal{M}_T, \mathcal{E}_T)$ whose vertices are matrices of size $|\mathcal{L}| \times T$, a *weighted scheduling graph* $(\mathcal{M}_T, \mathcal{E}_T, w_{\mathbf{a}})$ is a directed, weighted graph defined as follows: the vertex set is still $\mathcal{M}_T$, and each edge is associated with a weight. For a directed edge (v_1, v_2) in $(\mathcal{M}_T, \mathcal{E}_T)$, there is a weighed, directed edge (v_1, v_2) in $(\mathcal{M}_T, \mathcal{E}_T, w_{\mathbf{a}})$ with weight $w_{\mathbf{a}}(v_1, v_2) = \mathbf{a}^\intercal v_2 \mathbf{1}$, where $\mathbf{1} = [1, \ldots, 1]^\intercal \in \mathbb{R}^T$.

Since each achievable rate vector can be achieved by a periodic, collision-free schedule, which corresponds to a cycle in $(\mathcal{M}_T, \mathcal{E}_T)$ [7], we have the following result.

Lemma 2. *For a weighted scheduling graph $(\mathcal{M}_T, \mathcal{E}_T, w_{\mathbf{a}})$ and its maximum-mean-cycle $\mathcal{C} = (v_0, v_1, \ldots, v_m)$ with $m \geq 0$ and $v_0 = v_m$, the concatenation of the vertices in $\mathcal{C}$ gives a periodic schedule S' such that $R_{S'} \in \arg\max_{R\in\mathcal{R}} \langle \mathbf{a}, R\rangle$.*

Proof. For any $R \in \mathcal{R}$, suppose it is achieved by a schedule S that corresponds to a cycle $(v_0, v_1, \ldots, v_m)$ in $(\mathcal{M}_T, \mathcal{E}_T)$, by the definition of rate vectors (3). Given $\mathbf{a} = (a_i)_{i=1,\ldots,|\mathcal{L}|}$,

$$\begin{aligned}
\langle \mathbf{a}, R\rangle &= \sum_{i=1}^{|\mathcal{L}|} a_i R(l_i) \\
&= \sum_{i=1}^{|\mathcal{L}|} a_i \cdot \frac{1}{mT} \sum_{t=0}^{mT-1} S(l_i, t) \\
&= \frac{1}{mT} \sum_{i=1}^{|\mathcal{L}|} \sum_{t=0}^{mT-1} a_i \cdot S(l_i, t) \\
&= \sum_{j=0}^{m-1} \left(\frac{1}{mT} \sum_{i=1}^{|\mathcal{L}|} \sum_{k=0}^{T-1} a_i \left(v_j(i,k)\right) \right) \\
&= \frac{1}{mT} \sum_{j=0}^{m-1} \mathbf{a}^\intercal v_j \mathbf{1},
\end{aligned}$$

which states that $\langle \mathbf{a}, R\rangle$ equals to the average over values $\mathbf{a}^\intercal v_j \mathbf{1}$ for $j = 0, 1, \ldots, m-1$. □

Therefore, given a vector $\mathbf{a}$, finding a vector that solves (9) is equivalent to finding the maximum-mean-cycle in $(\mathcal{M}_T, \mathcal{E}_T, w_{\mathbf{a}})$, which is a widely studied problem [33], [34] that can be solved with time complexity $\Theta(nm)$, where n is the number of nodes and m is the number of edges in the graph. A classical algorithm is the Karp's algorithm [33], which is briefly reviewed below for the sake of completeness.

Given the graph with vertex set $\mathcal{V}$ and a source node $s \in \mathcal{V}$, for each $v \in \mathcal{V}$ and every non-negative integer k, suppose $F_k(v)$ denotes the maximum weight of a length-k path from s to v, and we say $F_k(v) = -\infty$ if such a path does not exist. Then the maximum cycle mean λ^* can be derived from the following theorem, whose proof can be found in [33].

Algorithm 2 Algorithm for Networks with Delays

Input: a network $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{I}, D)$
Output: maximum multiflow v

Start with any rate vector $R_1 \in \mathcal{R}$, $\mathcal{R}_1 \leftarrow \{R_1\}$
Construct $(\mathcal{M}_T, \mathcal{E}_T)$ from $\mathcal{N}$
for $i = 1, 2, \ldots$ **do**
  Run $v_i \leftarrow$LP-MMF($\mathcal{R}_i$) (or $v_i \leftarrow$LP-MCMF($\mathcal{R}_i$)) and obtain the dual vector μ_i
  Construct $(\mathcal{M}_T, \mathcal{E}_T, w_{\mu_i})$ by $(\mathcal{M}_T, \mathcal{E}_T)$ and μ_i
  Find maximum-mean-cycle in $(\mathcal{M}_T, \mathcal{E}_T, w_{\mu_i})$ and obtain $R_{i+1} \leftarrow \arg\max_{R \in \mathcal{R}} \langle \mu_i, R \rangle$
  $\mathcal{R}_{i+1} \leftarrow \text{conv}\,(\mathcal{R}_i \cup \{R_{i+1}\})$
  if $\langle \mu_i, R_{i+1} \rangle = \max_{R \in \mathcal{R}_i} \langle \mu_i, R \rangle$ **then**
    return v_i

Note we assume the graph $(\mathcal{M}_T, \mathcal{E}_T)$ is strongly connected, and hence $(\mathcal{M}_T, \mathcal{E}_T, w_{\mathbf{a}})$ is also strongly connected. Otherwise, we find the strongly connected components (with linear time complexity), search for the maximum-mean-cycle in each component and choose the one with the largest cycle mean.

Theorem 3 (Karp's Theorem [33])**.** *Given a strongly connected graph, the maximum cycle mean λ^* is given by*

$$\lambda^* = \max_{v \in \mathcal{V}} \min_{0 \le k \le n-1} \frac{F_n(v) - F_k(v)}{n - k}, \tag{13}$$

where $\mathcal{V}$ is the vertex set of the graph.

$F_k(v)$ can be given by a recurrence relation in [33], where $\mathcal{E}$ denote the edge set and $w(u, v)$ denotes the weight on (u, v):

$$F_k(v) = \max_{(u,v) \in \mathcal{E}} [F_{k-1}(u) + w(u, v)], \qquad k = 1, 2, \ldots, n$$

with the initial conditions $F_0(s) = 0$ and $F_0(v) = -\infty$, $v \neq s$. The Karp's algorithm computes $F_k(v)$ recurrently for $k = 0, 1, \ldots, n$ and $v \in \mathcal{V}$. For more related discussions, we refer the readers to [33], [34]. Finally, we present our Algorithm 2.

Theorem 4 (Optimality)**.** *For a network $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{I}, D)$, Algorithm 2 will terminate and output the maximum multiflow (or the maximum concurrent multiflow).*

The proof is similar to the proof of Theorem 1, though we use Lemma 2 to justify that in iteration i' when it terminates, the maximum-mean-cycle in $(\mathcal{M}_T, \mathcal{E}_T, \mu_{i'})$ gives

$$R_{i'+1} = \arg\max_{R \in \mathcal{R}} \langle \mu_{i'}, R \rangle,$$

and then we substitute it into

$$\langle \mu_{i'}, R_{i'+1} \rangle = \max_{R \in \mathcal{R}_{i'}} \langle \mu_{i'}, R \rangle,$$

which gives us the same result with (11). The remaining steps are similar with the proof of Theorem 1 and hence are omitted.

Remark 5. Karp's algorithm can find one maximum-mean-cycle in a time complexity polynomial in $|\mathcal{E}_T|$, the number of edges in the weighted scheduling graph. Nevertheless, $|\mathcal{E}_T|$ is exponential in $|\mathcal{L}|$, the number of links in the communication network. Therefore, the overall running time is at least exponential in $|\mathcal{L}|$, since we still desire the exact-optimal results. However, we note that finding the entire scheduling rate region generally has a time complexity exponential in $|\mathcal{E}_T|$, which should be doubly exponential in $|\mathcal{L}|$. As demonstrated in experiments in Section V, our algorithms can be significantly faster than the two-step method for both MMF and MCMF problems even in very simple networks, since we do not need the entire scheduling rate region.

We use $\mathcal{N}_{4,1}^{D=1}$ to illustrate our algorithm, which was also used in [7], [8] in particular to illustrate their scheduling scheme. However, we focus on the weighted scheduling graph and the maximum-mean-cycle scheme, and our Algorithm 2 based on them, which are not discussed in [7], [8].

Example 3. We use $\mathcal{N}_{4,1}^{D=1}$ as an example (see Figure 1). To first construct the scheduling graph, we find $T = \max_{l \in \mathcal{L}} \max_{l' \in \mathcal{I}(l)} |D(l, l')| = 1$ and denote the scheduling graph as $(\mathcal{M}_1, \mathcal{E}_1)$. The vertex set includes matrices of size 4×1, each of which is a column of some collision-free schedules.

We have a vertex set $\mathcal{M}_1$ $\{v_0, v_1, \ldots, v_8\}$, where

$$v_0 = \begin{bmatrix}0\\0\\0\\0\end{bmatrix}, v_1 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, v_2 = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, v_3 = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, v_4 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix},$$
$$v_5 = \begin{bmatrix}1\\0\\0\\1\end{bmatrix}, v_6 = \begin{bmatrix}1\\1\\0\\0\end{bmatrix}, v_7 = \begin{bmatrix}0\\1\\1\\0\end{bmatrix}, v_8 = \begin{bmatrix}0\\0\\1\\1\end{bmatrix}. \tag{14}$$

By implementing the algorithms in [7], [8] we can find the scheduling rate region $\mathcal{R}$, whose vertices are:

$$\begin{bmatrix}1/2\\1/2\\1/2\\1/2\end{bmatrix}, \begin{bmatrix}1/2\\1/2\\1/2\\0\end{bmatrix}, \begin{bmatrix}0\\1/2\\1/2\\1/2\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\\1\end{bmatrix}. \tag{15}$$

Then we show that we can find the maximum flow without the need of the entire scheduling rate region (15).

Suppose we start with a rate vector $R_1 = \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}^\intercal$ and in the first iteration we solve the following linear program

$$\begin{aligned} \max \quad & v = R(l_1) = R(l_2) = R(l_3) = R(l_4) \\ \text{s.t. } & R(l_i) \le 0 \qquad i = 1, 3, 4 \\ & R(l_2) \le 1 \\ & R(l_i) \ge 0 \qquad i = 1, 2, 3, 4 \end{aligned}$$

We obtain the maximum flow $v = 0$ and the corresponding dual vector is $\mu_1 = \begin{bmatrix}\epsilon & 0 & \epsilon & 0\end{bmatrix}^\intercal$ with some $\epsilon > 0$. We then use μ_1 to construct the corresponding weighted scheduling graph. Consider the vertex set (14), for $i = 0, 1, \ldots, 8$, the weights on the edges of the weighted scheduling graph are:

$$\begin{aligned} &w(v_i, v_0) = 0, \ \ w(v_i, v_1) = \epsilon, \ \ w(v_i, v_2) = 0, \\ &w(v_i, v_3) = \epsilon, \ \ w(v_i, v_4) = 0, \ \ w(v_i, v_5) = \epsilon, \\ &w(v_i, v_6) = \epsilon, \ \ w(v_i, v_7) = \epsilon, \ \ w(v_i, v_8) = \epsilon. \end{aligned}$$

We can verify that there are 9 vertices and 56 edges. By the maximum-mean-cycle algorithm, we solve $\arg\max_{R\in\mathcal{R}}\langle\mu_1, R_1\rangle$ and find $R_2 = \left[\frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2}\right]^\intercal$. We can check that the algorithm terminates in the next step.

Compared to the vertices of $\mathcal{R}$ shown in (15), our approach only needs to find two of them. The initial starting point (R_1) is easy to find by searching an arbitrary maximal independent set of $\mathcal{N}_{4,1}^{D=1}$ (shown in Figure 1). Moreover, we note that the maximum flow value $1/2$ indeed achieves the upper bound of line networks with utilizing nonzero delays, as proved in [9].

V. Performance Evaluation

In this section, we numerically compare our algorithms to conventional approaches, across various network settings [1]:

- We evaluate Algorithm 1 on random networks;
- We evaluate Algorithm 2 on different line networks that have also been used and evaluated in [7]–[9].

For all the experiments, we compare our algorithms with the two-step method (i.e., first calculate the whole scheduling rate region and then solve the MMF or MCMF problem), which is the conventional way to find exact-optimal solutions, although the method of calculating the scheduling region varies. Note that Algorithm 2 can also be evaluated on random networks, but we choose to perform experiments on different line networks because: 1) they have been discussed in literature [7]–[10], [26]; 2) due to the complexity of the scheduling algorithms [7], [8], the two-step method is unable to handle large-size networks; 3) we show that our algorithms have significant improvements even in very simple settings.

A. Algorithm 1 on Random Networks

We first evaluate Algorithm 1 on networks of N ($3 \leq N \leq 80$) nodes with random topology. As discussed in Section III, we let the networks be acyclic. For a given number of nodes, we randomly generate connected networks. We assume each link has a unit capacity, and we still use the 1-hop interference model. We study *multicast* case (in comparison, existing works may only work on unicast case [1], [3], [12]). We randomly choose 1 node as the source and 2 different nodes as the sink nodes. We do not specify any network coding mechanisms in particular, any network coding scheme can be directly applied (e.g., similar to [12] we can utilize the COPE system [22]). The main objective is to show the advantages of only calculating a subset of the scheduling rate region, while the conventional two-step method needs to calculate the scheduling region by searching maximal independent sets in the link conflict graph [1] before sovling the MMF problem.

We compare the required time to solve the MMF problem by the two approaches. In Figure 2, the required time (in logarithmic scale) is plotted against $3 \leq N \leq 80$, the sizes of the random networks. Note for each N, we randomly generate an N-node network for 5 times, which may have different topology. We randomly choose the source and sink nodes, and measure the average time used. Note the comparison is between the time used by two methods to find the optimal MMF values, and therefore the reduction in search time by our algorithm does not sacrifice any scheduling performance.

[1]All the implementations are based on Python, and executed on a laptop computer with i7-8550u CPU with Python 3.7.

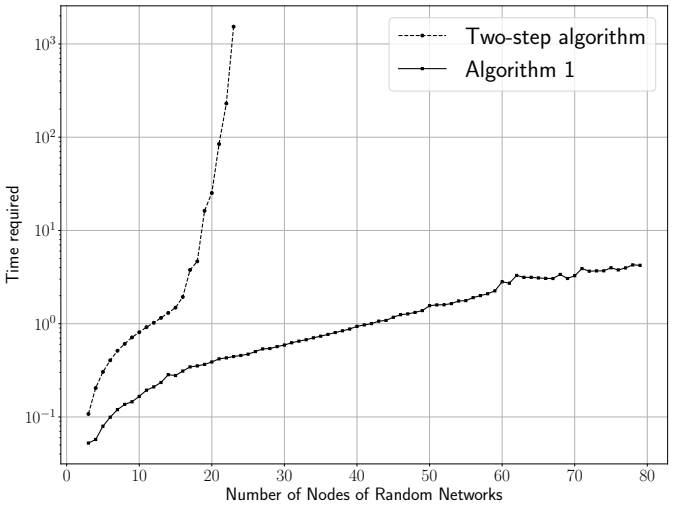

Fig. 2. The x-axis is the number of nodes in random networks. The y-axis (in logarithmic scale) is the time (seconds) to calculate the MMF value.

Figure 2 shows that our algorithm consistently outperform the two-step method, and it can be significantly faster when the network size becomes large. When the network size is large, the scheduling region (which is a polytope) not only becomes hard to compute, but also leads to two difficulties: 1) the rate vectors (the vertices of the polytope) are calculated in the form of vectors, but to iteratively update the scheduling region, we need to convert the set of vectors (V-representation) to a set of inequalities (H-representation), and the conversion suffers high complexity; 2) the termination condition of the algorithm needs to decide whether a vector achieves an optimum, which can also have high cost when the scheduling region is large. Terminating in a small number of steps is important (which is the main advantage of our algorithms), since the update of a subset of the scheduling region is a critical bottleneck and should be performed as little as possible.

B. Algorithm 2 on Line Networks

Next, we evaluate Algorithm 2 on various line networks with non-negligible delays [7]–[10] to show that not only do we outperform the conventional method based on [7], [8] for the scheduling problem, but also that our algorithm can be significantly faster even in simple networks. Due to the hardness of the scheduling problem with non-negligible delays [7]–[10], the two-step solution can only handle networks of small sizes. We choose $D = 1$ for simplification. We consider three tasks: single flow, MMF and MCMF maximizations:

1) **Single flow maximization**: Consider a unicast line network under the 1-hop interference model $\mathcal{N}_{L,1}^{D=1}$. The first node is the source, and the last node is the only sink.
2) **MMF problem**: Consider bi-directional line networks under the *single collision domain* model [10], [26], i.e.,

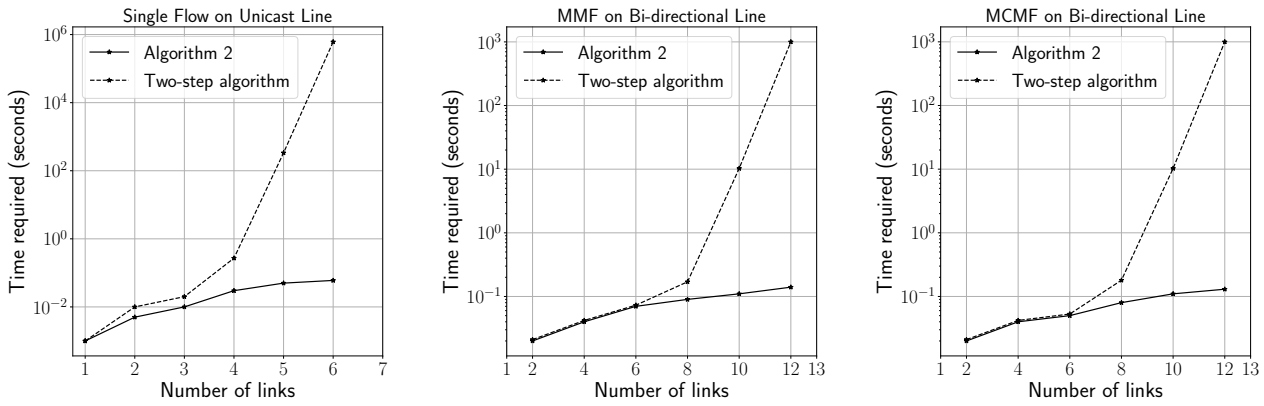

Fig. 3. The x-axis is the number of links and the y-axis (in logarithmic scale) is the required time (seconds) to calculate the maximum (concurrent) multiflow value. The left figure is for single-flow maximization in unicast line network $\mathcal{N}_{L,K}^{D=1}$; the middle figure is for the MMF problem in bi-directional line networks; the right figure is for the MCMF problem in bi-directional line networks.

each link can collide with all the other links. There are two information flows in the network, one from node 1 to node N, and another from node N to node 1.

3) **MCMF problem**: We use the same bi-directional line networks with the same interference model. For the two flows, we require that the desired traffic rate of one flow is half of the desired traffic rate of the other flow.

Similar to Section V-A, we compare our algorithm with the two-step method. However, instead of enumerating the maximal independent sets in the conflict graph, we use the cycle-enumeration method [7], [8] to calculate the scheduling rate region in the two-step method.

The performance evaluation is shown in Figure 3. The running time (in logarithmic scale) is plotted against the number of links L. The figure demonstrates that the time required by our algorithms to achieve the optimal (concurrent) multiflow values is significantly lower. Since the comparison is between the time required to find the optimal MMF and MCMF values, the reduction in search time by our algorithm does not compromise scheduling performance. Additionally, it is evident that due to the NP-hardness of the scheduling problem, the required time for solving the multiflow problem by the two-step scheme in networks with non-negligible delays increases doubly-exponentially fast, which aligns with theoretical studies in [7] and shows that the scheduling region is the key bottleneck, and hence we can have significant gains by only calculating a subset of the scheduling region.

For example, the algorithm in [7] needs to search 7653 rate vectors to characterize the scheduling region of $\mathcal{N}_{4,1}^{D=1}$, which has 9 vertices as shown in (15), while we only need to find $\mathbf{2}$ of them for solving the MMF (or MCMF) problem. For $\mathcal{N}_{6,1}^{D=1}$ whose rate region has 57 vertices, we only need $\mathbf{4}$ of them. The two-step method requires more than a week to calculate the scheduling rate region of $\mathcal{N}_{6,1}^{D=1}$, while Algorithm 2 can solve the MMF problem in less than **1 second**. In more complicated settings (e.g., random networks as discussed in Section V-A), the scheduling problem becomes even harder, which might make our joint approach more preferable.

VI. Conclusion

In large-scale wireless systems, although the maximum (concurrent) multiflow problem is extremely important for understanding the network capability, it is NP-hard and even computationally prohibitive to solve due to the hardness of the scheduling problem. It can become even harder if the propagation delays are considered in challenging network environments (e.g., underwater and deep-space). Most (distributed) linear programming algorithms with low complexities either provide approximated solutions or are designed for specific networks under certain constraints In this paper, we jointly solve the MMF and MCMF problems and the scheduling problem, in a general multi-source multi-sink network with network coding allowed and propagation delays potentially utilized in scheduling. We provide practical algorithms that may only need a small subset of the scheduling rate region for the MMF and MCMF problems, making our approaches more efficient without sacrificing solution accuracy or scheduling performance. We prove that our algorithms output exact-optimal solutions in a finite number of iterations, and experiments show that our algorithms can be significantly faster.

VII. Future Works

In Section IV-B, we utilize the maximum-mean-cycle algorithms in solving the MMF (or MCMF) problem. We find thatthese algorithms can also be used to calculate the throughput (the objective in [9], [10]) or the entire scheduling rate region (the objective in [7], [8]) as follows. The convex hull method [35] is an algorithm for finding the vertices of an unknown polytope $\mathcal{P} \subseteq \mathbb{R}^n$, given an oracle that can find $\operatorname{argmax}_{x\in\mathcal{P}}\langle x, a\rangle$ for $a \in \mathbb{R}^n$. By using the maximum-mean-cycle algorithm as the oracle to the convex hull method, we can iteratively search for the vertices of the entire scheduling rate region. This method can be more efficient than the cycle-enumeration method in $(\mathcal{M}_T, \mathcal{E}_T)$ [7], since we do not need to enumerate all the simple cycles in $(\mathcal{M}_T, \mathcal{E}_T)$. Detailed implementations of such algorithms for the scheduling problem is left for future study.

REFERENCES

[1] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless networks*, vol. 11, no. 4, pp. 471–487, 2005.

[2] B. S. Baker, "Approximation algorithms for np-complete problems on planar graphs," *Journal of the ACM (JACM)*, vol. 41, no. 1, pp. 153–180, 1994.

[3] P.-J. Wan, "Multiflows in multihop wireless networks," in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, 2009, pp. 85–94.

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on information theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[5] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE transactions on information theory*, vol. 49, no. 2, pp. 371–381, 2003.

[6] R. W. Yeung, *Information theory and network coding*. Springer Science & Business Media, 2008.

[7] J. Ma, Y. Liu, and S. Yang, "Rate region of scheduling a wireless network with discrete propagation delays," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[8] S. Yang, J. Ma, and Y. Liu, "Wireless network scheduling with discrete propagation delays: Theorems and algorithms," *arXiv 2107.03083*, 2023.

[9] W. Bai, M. Motani, and H. Wang, "On the throughput of linear unicast underwater networks," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[10] M. Chitre, M. Motani, and S. Shahabudeen, "Throughput of networks with large propagation delays," *IEEE J. Ocean. Eng.*, vol. 37, no. 4, pp. 645–658, Jul 2012.

[11] C.-C. Hsu, K.-F. Lai, C.-F. Chou, and K.-J. Lin, "ST-MAC: Spatial-temporal MAC scheduling for underwater sensor networks," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 1827–1835.

[12] J. Zhou, S. Xia, Y. Jiang, H. Zheng, and L. Cui, "Maximum multiflow in wireless network coding," *IEICE transactions on communications*, vol. 96, no. 7, pp. 1780–1790, 2013.

[13] N. M. Jones, B. Shrader, and E. Modiano, "Optimal routing and scheduling for a simple network coding scheme," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 352–360.

[14] T. Cui, L. Chen, and T. Ho, "Distributed optimization in wireless networks using broadcast advantage," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 5839–5844.

[15] T. Wiese, M. Riemensberger, and W. Utschick, "Scheduling for network-coded multicast with interference," *IEEE Transactions on Signal Processing*, vol. 64, no. 9, pp. 2245–2254, 2016.

[16] R. Niati, A. H. Banihashemi, and T. Kunz, "Throughput and energy optimization in wireless networks: Joint mac scheduling and network coding," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 3, pp. 1372–1382, 2012.

[17] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *Journal of the ACM (JACM)*, vol. 37, no. 2, pp. 318–334, 1990.

[18] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003, pp. 42–54.

[19] ——, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *Proceedings of the 11th annual international conference on Mobile computing and networking*, 2005, pp. 73–87.

[20] V. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "Algorithmic aspects of capacity in wireless networks," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2005, pp. 133–144.

[21] H. Su and X. Zhang, "Characterizing the throughput gain of network coding in wireless ad hoc networks," in *MILCOM 2008-2008 IEEE Military Communications Conference*. IEEE, 2008, pp. 1–7.

[22] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, 2006, pp. 243–254.

[23] X. Yan, R. W. Yeung, and Z. Zhang, "An implicit characterization of the achievable rate region for acyclic multisource multisink network coding," *IEEE Transactions on Information Theory*, vol. 58, no. 9, pp. 5625–5639, 2012.

[24] D. Traskov, M. Heindlmaier, M. Médard, and R. Koetter, "Scheduling for network-coded multicast," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1479–1488, Oct. 2012.

[25] J. G. Proakis, "Intersymbol interference in digital communication systems," *Wiley Encyclopedia of Telecommunications*, 2003.

[26] Y. Fan, Y. Liu, and S. Yang, "Continuity of link scheduling rate region for wireless networks with propagation delays," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 730–735.

[27] Y. Guan, C.-C. Shen, and J. Yackoski, "MAC scheduling for high throughput underwater acoustic networks," in *Proc. IEEE WCNC 2011*, Cancún, Mexico, Mar 2011, pp. 197–202.

[28] A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," in *Proc. Workshop on Network Coding, Theory & Applications*. Citeseer, 2007, pp. 1–9.

[29] M. Yang and Y. Yang, "A linear inter-session network coding scheme for multicast," in *2008 Seventh IEEE International Symposium on Network Computing and Applications*. IEEE, 2008, pp. 177–184.

[30] M. Langberg and A. Sprintson, "On the hardness of approximating the network coding capacity," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 1008–1014, 2011.

[31] C. T. Li, "Undecidability of network coding, conditional information inequalities, and conditional independence implication," *IEEE Transactions on Information Theory*, vol. 69, no. 6, pp. 3493–3510, 2023.

[32] P. M. Gleiss, J. Leydold, and P. F. Stadler, "Circuit bases of strongly connected digraphs," 2001.

[33] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete mathematics*, vol. 23, no. 3, pp. 309–311, 1978.

[34] A. Dasdan and R. K. Gupta, "Faster maximum and minimum mean cycle algorithms for system-performance analysis," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 17, no. 10, pp. 889–899, 1998.

[35] C. Lassez, "Quantifier elimination for conjunctions of linear constraints via a convex hull algorithm," *Symbolic and Numerical Computation for Artificial Intelligence*, pp. 103–122, 1992.