IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 20, 2020, accepted February 2, 2020, date of publication February 6, 2020, date of current version February 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2972110

Joint BATS Code and Periodic Scheduling in Multihop Wireless Networks

ZHIHENG ZHOU, (Member, IEEE), JINGMIN KANG, AND LIANG ZHOU
National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu 610051, China
Corresponding author: Liang Zhou (lzhou@uestc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61801095.

ABSTRACT Batched sparse (BATS) code is a promising technology for reliable data transmission in error-prone environments. A BATS code consists of an outer code and an inner code. It has been shown that a well-designed inner code is critical to the performance of BATS. Though great efforts have been made in the design of inner codes, the impact of scheduling (that determines when inner codes take place at nodes in the path) on inner codes is still unclear. In this paper, we study the joint design of inner code and scheduling in multihop wireless networks. We first introduce a new network utility from which to associate inner codes with scheduling, and formulate the coding constraint and the scheduling constraint using independent sets of a graph. With the utility, the joint design problem is then transformed to a network utility maximization problem under the constraints. We next exploit the property of the proposed optimization problem and reveal the relation between the expected batch transfer matrix rank and maximal independent sets. In the light of their relationship, we propose joint coding and scheduling rules and show that a periodic scheduling can be used to achieve provable performance guarantees. However, under most realistic coding settings, the proposed coding and scheduling problem is NP-hard. In order to meet the practical needs of the implementation, we develop greedy algorithms that attempt to iteratively improve the current best solution. Numerical results show that our algorithms enables us to approach the utilization of multi-hop wireless networks with a relatively low end-to-end delay.

INDEX TERMS Network coding, batched sparse code, network utility maximization, scheduling.

I. INTRODUCTION

Multi-hop wireless networks are emerging in many applications, such as wireless sensor networks, underwater networks and vehicular networks. In these scenarios, mission-critical tasks may require a source node to reliably transmit data to a destination node via multiple relay nodes along a single path. However, wireless communications experience severe packet loss due to the multipath effect, interference, limited resources and hidden nodes. The more the number of hops is, the higher the packet loss probability [1]. In order to provide the end-to-end reliability in multi-hop wireless networks, various techniques, such as retransmission [2], network coding [3]–[5] and fountain codes [6], [7], have been proposed to facilitate the packet loss issue. However, these techniques are not efficient for multihop wireless networks due to high complexity, low throughput and/or long delay [11], [12].

BATched Sparse (BATS) codes [8], as a promising new technique, were proposed to achieve the reliable communications between end nodes. A BATS code is composed of an *outer code* and an *inner code*. The outer code is a matrix generalized fountain code that generates a potentially unlimited number of batches.[1] Each batch consists of M coded packets, called *batch size*. The inner code (also called *recoding*) is a random linear network coding (RLNC cf. [4]) applied on packets belonging to the same batch, performed at nodes that transmit batches. The outer code and the inner code can be jointly decoded at the destination node using a low computational complexity algorithm, e.g., Belief Propagation (BP).

BATS codes take the advantages of both fountain codes and RLNC. BATS codes preserve the salient feature of fountain codes, in particular, their low encoding/decoding complexity and the ratelessness. Thus, BATS codes is able to approach the performance of the ordinary RLNC schemes [8] using a mild batch size, resulting in the low coding overhead and the small storage requirements at relay nodes. These features

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong.

[1] When $M = 1$, the outer code becomes an LT code (or a Raptor code if precoding is applied).

make BATS codes suitable for multihop wireless networks. Afterwards, many works have been concerned with the implementation of BATS codes in a variety of wireless applications, such as wireless sensor networks [10], vehicular networks [17], underwater acoustic networks [12], Internet of Things (IoT) [13], etc..

For BATS codes, a well-designed inner code is proven to be a crucial part impacting the transmission performance such as throughput. In particular, Tang *et al.* [14] and Yin *et al.* [15] indicated that doing recoding without the consideration of channel conditions is not optimal for the end-to-end throughput in line networks, and provided an inner code that can adapt to the packet loss rates on transmission links.

Though Tang and Yin's work show a significant gain in throughput, none of them discuss the scheduling problem of determining when communication takes place between the nodes in the path. The scheduling problem is particularly essential for multihop wireless networks due to the conflicts between neighboring nodes [22]. For example, in a wireless network of nodes with identical and omnidirectional radio ranges, the conflict dictates that only one of the 2 hops can be active at a time. Therefore, the design of inner codes for multihop wireless networks must be subject to the certain scheduling policy.

On the other hand, BATS codes brings new challenges for scheduling, particularly in the context of the use of RLNC in inner codes. One problem is that the change of a queue cannot be simply measured by differentiating the number of sent and received packets. The reason is that the number of recoded packets is larger than of received packets. Besides, the number of recoded packets for a batch depends on channel conditions and thus may be different at different nodes [14], [15], [20]. The last problem is that the transmission efficiency for BATS is a non-linear function of the number of received packets. Thus, the solutions for existing network optimization problems may not be suitable for BATS. In order to tackle these problems, a scheduling policy should be design to accommodate the characteristics of inner codes.

Driven by the above observations, the joint design of inner codes and scheduling is the primary focus of this paper. As a result, our main objective is to study the relation between inner codes and scheduling and catch the constraints between them. However, the corresponding optimization problem is combinatorial with complex practical constraints, which in general is difficult to be solved optimally. The secondary objective is then to develop an efficient algorithm that can yield a near-optimal solution yet with low computational complexity. Towards these end, the major contributions of this paper are summarised as follows.

- We introduce a new network utility, which is a quantitative measure for the ability of an inner code to support a schedule plan. The joint design of inner codes and scheduling is thus modeled as a network utility maximization problem.
- We consider a scheduling as a sequence of independent sets and formally describe the relation between the empirical rank distribution and the sequence. Sequentially, this relationship is presented as the joint recoding-scheduling rules. To the best of our knowledge, this is the first attempt to connect BATS codes with the scheduling problem.
- We construct a NonLinear Integer Programming (NLIP) problem to meet the both practical and performance needs. We further prove that a scheduling built by a finite length sequence can arbitrarily approach the optimal performance yet with much lower scheduling overhead.
- We develop a simple gradient search algorithm to efficiently solve the NLIP problem. The algorithm is an iterative procedure taking steps in the most effective direction of improving the network utilization.
- We propose an ordering algorithm that organizes the independent sets in an ascending order with respect to the distance from nodes to the source, and thus can reduce the end-to-end delay.
- Numerical results show the efficiency of our algorithms in terms of the network utilization and the end-to-end delay.

The rest of the paper is organized as follows. In Section II, we briefly introduce BATS codes and related works. In Section III, we presents the network model and needed definitions. In Section IV, we formulate a network utility maximization problem and develop the joint solution of recoding and scheduling. The efficient approaches are proposed to find the near-optimal solutions in term of network utility and end-to-end delay, respectively. In Section V, we provide the numerical evaluations. Finally, in Section VI, We conclude with a summary of our results.

II. BATS CODES AND RELATED WORK

A. BATS CODES BASICS

In this subsection, we give a brief view on BATS codes. Readers may find a detailed discussion of BATS codes in [9].

A BATS code comprises of an outer code and an inner code. Suppose a source node needs to send K packets to a destination node via a wireless network, where each symbol in packets is an element of the finite field $\mathbb{F}_q$ with q elements. Fix an integer $M \geq 1$ as the batch size. Using the outer code, a sequence of batches $\mathbf{X}_i, i = 1, 2 \ldots$ are generated as,

$$\mathbf{X}_i = \mathbf{B}_i \mathbf{G}_i,$$

where $\mathbf{B}_i$ is a matrix consisting of dg_i columns, each of which is a source packet that is randomly picked out, and $\mathbf{G}_i$ is a *totally random matrix* on $\mathbb{F}_q$ of dimension $\mathrm{dg}_i \times M$. Here, dg_i is called the *degree* of the i-th batch $\mathbf{X}_i$. The degrees $\mathrm{dg}_i, i = 1, 2, \ldots,$ are i.i.d. random variables with a degree distribution.

Before forwarding a received batch, the network nodes will first apply (random) linear network coding, i.e., recoding, on packets belonging to the same batch to generate a number

of recoded packets for transmitting. The source node perform recoding on each batch to generate a number (which is what we investigate in the paper) of packets for transmitting. Moreover, each intermediate node will apply the same recoding process on all the received batches. The recoding at the source node and the intermediate nodes forms the inner code of the BATS code.

As recoding is a linear operation within a batch, the received packets of the i-th batch, denoted by $\mathbf{Y}_i$, can be expressed as

$$\mathbf{Y}_i = \mathbf{X}_i\mathbf{H}_i = \mathbf{B}_i\mathbf{G}_i\mathbf{H}_i, \tag{1}$$

where $\mathbf{H}_i$ is called the *(batch) transfer matrix*. At the destination node, the linear equations in (1) for $i = 1, \ldots$ are used to recover the K source packets, which can be solved by an efficient BP decoding algorithm.

B. RELATED WORK

The design of the outer code and the inner code can be largely separated given the destination node transfer matrix rank distribution (called the rank distribution henceforth). The design of the outer code has been extensively studied in [8], [18], [19]. In [8], a sufficient condition for the BP decoder to recover a given fraction of the input packets with high probability was obtained. It has been verified theoretically for certain special cases and demonstrated numerically for general cases that BATS codes can achieve rates very close to the expected rank of the given rank distribution of the transfer matrices, which is an upper bound on the achievable rate of BATS codes. In [18], the finite-length performance of the outer code has been studied and showed that inactivation decoding can achieve rates very close to the expected rank of the given rank distribution of the transfer matrices when K is relatively small.

Based on the performance of the outer code, the main goal of the design of the inner code is to maximize the expected rank of the transfer matrices normalized by certain network communication cost. Suppose that the network has l links and the i-th link is used t_i times during the transmission. In the study of the inner code design in [14], [15], the communication cost is the maximum uses, $\max_{i=1}^{l} t_i$, of all the network links, and the corresponding performance measure is called *throughput*. In [20], we study the inner code design using the communication cost $\sum_{i=1}^{l} t_i$ to maximize the energy efficiency. The same cost is utilized in [21] to minimize the average number of recoded packets among all the batches in a relay-assisted network. These works indicate that choosing a *communication cost* is critical to the design of inner codes. In this paper, one of our goal is to find a proper communication cost as the bridge between inner codes and scheduling.

On the other hand, there have been many studies on the implementation of BATS codes in various wireless applications. For example, the study in [10] and [11] proposed a FUN framework, where an inner-encoding algorithm was designed to mix the packets belonging to two intersecting flows in TDMA multi-hop networks. In [16], the authors proposed a distributed two-phase cooperative broadcasting protocol, which uses BATS codes in the first phase to help the peer-to-peer (P2P) communications. Further, a BATS-based V2X communication scheme was developed, where a vehicle node determines which batch should be forwarded first according to its contribution to the transmission [17]. In [13], BATS was employed to improve the reliability of the uplink transmissions in industrial Internet of Things. In underwater acoustic multi-hop networks, the utilization of BATS under different channel conditions was evaluated [12]. Although these works show that using BATS can provide a better performance than the traditional approaches, they did not consider interference in wireless multi-hop networks. In this paper, we thus aim to investigate the effects of BATS and interference on each other and develop practical coding and scheduling approaches.

III. SYSTEM MODEL

A. NETWORK MODEL

Consider a flow traversing a wireless network through a single path. The path comprises of $n + 1$ nodes, denoted by $v_i, i = 1, 2, \ldots n + 1$, where v_1 is source and v_{n+1} is destination. Direct communication links exist only between two consecutive nodes. Let l_i denote the link connecting nodes v_i and v_{i+1}. We assume that each link has a fixed integer capacity c_i packets per unit time. The packets transmitted on a link is lost independently. Let ϵ_i be the packet loss rate on the links $l_i, i = 1, 2, \ldots, n$. We consider a time slotted setting, where multiple links may be activated at the beginning of a slot and serve packets without interference within the slot. The terms transmit and serve are used interchangeably in the following.

Let e denote an independent set, in which links can transmit simultaneously without interference with each other.[2] Let set $\mathcal{S}$ consist of all independent sets. Each independent set e corresponds to an n-dimensional rate vector $\mathbf{r}_e$, in which the i-th entry is

$$r_{e,i} = \begin{cases} c_i & i \in e, \\ 0 & otherwise. \end{cases} \tag{2}$$

At each slot, the scheduling policy will select a set e and allow the links in e to serve data at the rate in $\mathbf{r}_e$. The scheduling problem is thus to determine a sequence $\Upsilon_{\mathcal{S}}$ of independent sets in a fashion that meets the optimization goal for some network performance of interest, that is throughput in this paper.

B. RECODING

During the file transmission, the source node encodes K input packets using a BATS code of batch size M. For each batch

[2]We assume that there is a centralized scheduler in the network such as industrial wireless network [26], which guarantees that links not in an independent set cannot be activated simultaneously. In some wireless applications, e.g. vehicular networks, a centralized scheduler is hard to implement. For these cases, a distributed scheduling is desirable, e.g. [25]. We will investigate it in the next work.

generated by the outer code, the source node further recodes the batch by random linear coding to t_1 (> 0) packets. The source node then transmits t_1 packets of a batch in t_1 uses of link l_1.

The recoding at an intermediate node can be formulated inductively. Consider node v_i, $2 \leq i \leq n$. The intermediate node applies RLNC on a batch if it receives at least one packet for the batch. Otherwise, the node do noting for the batch. Particularly, node v_i will generate t_i packets and transmit these packets in t_i uses of link l_i. We call t_i the recoding size at node v_i.

As recoding on each batch are independent, we use one generic batch to formulate the recoding process. Let $\mathbf{X}_{in}$ be a matrix over finite field $\mathbf{F}_q$, where each column is a packet of a batch received by node v_i, $1 \leq i \leq l$. (For node v_1, $\mathbf{X}_{in}$ is the batch generated by the outer code.) Then the recoding at v_i can be expressed as

$$\mathbf{X}_{out} = \mathbf{X}_{in}\Phi_i, \tag{3}$$

where Φ_i is a t_i-column matrix consisting of coding coefficients independently and randomly selected from $\mathbb{F}_q$ and called recoding matrix.

Moreover, a transfer matrix can be fully described by means of Φ_i and ϵ_i, $i = 1, \ldots, n$. With slight abuse of notation, we use $\mathbf{H}_k$ to represent a transfer matrix obtained by node v_i. Define $\mathbf{D}_i$ as a $t_i \times t_i$ random diagonal matrix consisting of independent diagonal entries $d_{jj} = 1$ with probability $1 - \epsilon_i$ and $d_{jj} = 0$ with probability $\epsilon_i, j = 1, 2, \ldots, t_i$. The transfer matrix $\mathbf{H}_{i+1}$ can, then, be expressed as,

$$\mathbf{H}_{i+1} = \mathbf{H}_i\Phi_i\mathbf{D}_i, \quad i = 1, \ldots, n, \tag{4}$$

where $\mathbf{H}_1 = \text{diag}(1, 1, \ldots, 1)$ is an $M \times M$ identity matrix. In addition, we postulate that $\Phi_1, \ldots, \Phi_n, \mathbf{D}_1, \ldots, \mathbf{D}_n$ are mutually independent.

Let $\mathbf{h}_i = [h_{i,0}, h_{i,1}, \ldots, h_{k,M}]$ be the batch transfer matrix rank distribution at node v_i, where $h_{i,m}$ is the probability that a batch has a transfer matrix with rank m at node v_i, $1 \geq i \geq n$. In addition, let $\bar{h}_i = \sum_{m=0}^{M} m h_{i,m}$ be the expected rank of a transfer matrix. It is easily seen from (1) that $\bar{h}_{n+1}$ represents the transmission capacity of batches, i.e., the average number of innovative packets can be delivered to the destination by using one batch.

Note that, though BATS uses RLNC to recode batches, it does not require intermediate nodes to decode batches, instead the decoding process is only done at the destination. As a result, the ordinary RLNC can outperform BATS if both encoding and decoding are performed at every intermediate node, e.g. [25]. However, it brings high coding overhead. How to trade off between coding overhead and coding performance can be found in [9] but is not in the scope of this paper.

IV. MAIN RESULTS

In this paper, we aim to maximize end-to-end throughput of a given source and destination pair in a multi-hop wireless network by jointly designing inner codes of BATS and scheduling. Towards this end, the relation of throughput to both inner codes and scheduling is investigated and formulated as an utility maximization problem. Next, a joint recoding and scheduling policy is presented followed by discussions about the design of a scheduling using a finite sequence. At the end of this section, approximate algorithms are designed to efficiently solve the proposed optimization problem and reduce the end-to-end delay.

A. PROBLEM FORMULATION

Given rank distribution $\mathbf{h}_{n+1}$ at the destination node, it has been demonstrated that the outer code can achieve a rate, defined as K/N, very close to the expected rank $\bar{h}_{n+1}$, where K is the number of original packets and N is the number of batches transmitted at the source node [8]. Define *network utilization* η as the ratio of $\bar{h}_{n+1}$ to the expected time, denoted by T, required to transmit a batch from the source node to the destination node. The end-to-end throughput of the system is proportional to network utilization η [22]. In this paper, we then focus on maximizing the network utilization η.

From the previous work, we know that $\bar{h}_{n+1}$ can be derived from a function of $t_i, i = 1, 2, \ldots, n$, given batch size M, field size q and packet loss rates ϵ_i [9]. Let $\bar{h}_{n+1} = f(t_1, \ldots, t_n)$, where $f : \mathbb{N}^n \to \mathbb{R}$. Function f is strictly increasing with respect to the recoding sizes of nodes $v_i, i = 1, \ldots, n$. Maximizing $t_i, i = 1, 2, \ldots, n$, is thus one of our main concerns in maximizing η. However, the average serve rate of a link must be no smaller than the average arrival rate of packets at the sender [27]. Otherwise congestion occurs, resulting in a drop in throughput.

In our model, it requires that the average service rate offered to link l_i cannot be smaller than the average number of coded packets node v_i generates. Node v_i can recode a batch if it receives at least one packet of the batch. Since each recoded packets for a batch plays the same role, the batch is considered as lost at node v_i only if none of the t_{i-1} packets is successfully arrived at v_i. Let $Q_i, i = 1, 2, \ldots, n$, be the number of transmitted packets for a batch at link l_i. In the light of the recoding scheme described above, we know that $T_1 \to T_2 \to \cdots \to Q_n$ forms a Markov chain, and i) $\Pr\{Q_1 = t_1\} = 1$, ii) $\Pr\{Q_{i+1} = 0 | Q_i = t_i\} = \epsilon_i^{t_i}$ and $\Pr\{Q_{i+1} = t_{i+1} | Q_i = t_i\} = 1 - \epsilon_i^{t_i}$, iii) $\Pr\{Q_{i+1} = 0 | Q_i = 0\} = 1$. We derive that for $i = 1, \ldots, n$, $\mathbb{E}[Q_i] = t_i \prod_{j=1}^{i-1}(1 - \epsilon_j^{t_j})$. That is, given channel condition, the expected number of transmitted packets for a batch is a function of recoding sizes t_i that are what we investigate in the paper.

Next, let α_e denote the usage frequency of independent set e used in a scheduling sequence. We then obtain the following constraint between an inner code and a scheduling policy.

$$\sum_{e \in \mathcal{S}} \alpha_e r_{e,i} T \geq t_i \prod_{j=1}^{i-1}(1 - \epsilon_j^{t_j}), \quad \sum_{e \in \mathcal{S}} \alpha_e = 1, \ \alpha_e \geq 0, \tag{5}$$

where $r_{e,i}$ is the i-th entry of $\mathbf{r}_e \in \mathcal{R}$. Equation (5) is a necessary condition for network stability under any scheduling, i.e., the queue length at each node stays finite. Sequentially,

our optimization goal is to maximize network utilization η under the constraint (5).

On the other hand, it is natural to show that any independent set is a subset of a maximal independent set. A maximal independent set is an independent set that is not properly contained in any other independent set. This fact, combined with the above discussions, indicates that the scheduler should only select a solution from maximal independent sets at each time slot even if there is a single flow. For instance, suppose that a scheduling sequence $\Upsilon_{\mathcal{S}}$ contains a non-maximal independent set e. After replacing set e by its corresponding maximal independent set e_m, we construct a new sequence $\Upsilon'_{\mathcal{S}}$. Thus the average capacity assigned to links $l_i, i = 1, \ldots, n$, by sequence $\Upsilon'_{\mathcal{S}}$ is no less than those by $\Upsilon_{\mathcal{S}}$. Then equation (5) implies that the optimal recoding sizes for $\Upsilon'_{\mathcal{S}}$ is no less than these for $\Upsilon_{\mathcal{S}}$. That is, by means of maximal independent sets we can always obtain a better network utilization. Let set $\mathcal{S}_{max}$ consist of all maximal independent sets, and set $\mathcal{R}_{max}$ consist of all rate vectors $\mathbf{r}_e$ corresponding to $e \in \mathcal{S}_{max}$.

Finally, the network utilization maximization problem (P) is formulated as follows.

$$\begin{aligned}
&\underset{\substack{t_1,\ldots,t_n,T\\ \alpha_e, e\in\mathcal{S}_{max}}}{\text{maximize}} \ \frac{f(t_1,\ldots,t_n)}{T}\\
&\text{subject to } t_i \in \mathbb{N}, \quad t_i \geq M,\ i = 1,\ldots,n,\\
&\qquad \sum_{e\in\mathcal{S}_{max}} \alpha_e r_{e,i} T \geq t_i \prod_{j=1}^{i-1}(1-\epsilon_j^{t_j}),\\
&\qquad \alpha_e \geq 0, \quad \sum_{e\in\mathcal{S}_{max}} \alpha_e = 1,\ \mathbf{r}_e \in \mathcal{R}_{max}. \qquad \text{(P)}
\end{aligned}$$

Solving the problem (P) requires the explicit formulation of function f that is directly related to both the recoding and scheduling policies. In the next subsection, we will dive deeper into their relationship from which to investigate joint design of recoding and scheduling.

B. JOINT DESIGN OF INNER CODE AND SCHEDULING

The scheduling sequence $\Upsilon_{\mathcal{S}_{max}}$ does not only limit the number of packets node v_i can transmit, but also determines the number of received packets of a batch when node v_i tries to forward the batch. For inner codes described in Section III.B, this impacts the construction of matrix Φ in (3). Consider an example that node v_{i+1} receives two packets of a batch at slot t, while v_i also has some packets of the batch. If a scheduler decides that v_{i+1} transmits at $t+1$, v_i transmits at $t+2$, and v_{i+1} transmits at $t+3$, then the recoding matrix at v_{i+1} at $t+3$ is

$$\Phi = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ 0 & x_{32} \end{bmatrix} \tag{6}$$

where coding coefficients x_{ij} are independently and uniformly selected from a finite field, and we assume that v_{i+1} successfully receives a packet from v_i at $t+2$. On the other hand, consider another scheduler deciding that v_i transmits at t+1, v_{i+1} transmits at $t+2$, and v_{i+1} transmits at $t+3$, then the recoding matrix at v_{i+1} at $t+3$ is

$$\Phi' = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} \tag{7}$$

where we assume that v_{i+1} successfully receives a packet from v_i at $t+1$. The linear operator channel theory [23] shows that recoding as (7) can deliver more information than as (6). Note that, the performance of BATS is highly related to the expected rank of the transfer matrices (the relation between the recoding matrix and the transfer matrix is given in Section III.B). As a result, we derive the following proposition to prove that the second scheduling policy can improve BATS performance in term of the expected rank of the transfer matrices at the destination.

Proposition 1: For the inner codes described in Section III.B, given $t_i, \epsilon_i, i = 1, \ldots, n$, and q, larger expected rank $\bar{h}_{n+1}$ can be obtained by starting recoding on a batch after the previous hop stops transmitting the batch than before.

Proposition 1 suggests that in order to achieve the maximum network utilization, node v_i ($1 \leq i \leq n$) should start transmitting the first recoded packets for a batch only after v_{i-1} stops forwarding the batch. Next, let us show that there exits a near-optimal scheduling satisfying both the above recoding rule and constraint (5).

We notice that the transmission of a batch is most likely to be "error-free", i.e., a forwarded batch reaches the next hop almost surely. This is due to the fact that the probability of the loss of a batch at node v_i is equal to $1-\epsilon_{i-1}^{t_{i-1}}$. For typical batch sizes (e.g., $M \geq 8$), this probability is very close to 1. For example, if $M \geq 8$ and $\epsilon_i \leq 0.35$, then $1-\epsilon_i^{t_i} \geq 0.9997$, $1 \leq i \leq n$.

The aforementioned recoding rule and the "error-free" property motivate us to deliver batches in a periodic fashion using a fixed scheduling sequence, rather than to serve them randomly. First, we complete the joint recoding-scheduling policy as follows, where Rule 2 and 3 guarantee that the behaviour of nodes can keep consistent in each period, and Rule 4 enables the maximum use of channel capacity.

Joint Recoding-Scheduling Rules:

1) Nodes $v_i, i = 2, \ldots, n$, can start to recode and transmit a batch only after all the t_{i-1} packets of the batch have been transmitted by v_{i-1}.
2) Nodes $v_i, i = 2, \ldots, n$, transmit batches in a sequential order, i.e., a received batch can be transmitted only after t_i packets of every earlier received batch have already been transmitted.
3) The nodes send a packet only once regardless of whether the transmission is successful or not.
4) Even if t_i packets of the most recently arrived batch have already been transmitted, the node should recode and transmit recoded packets for the batch at an assigned time slot.

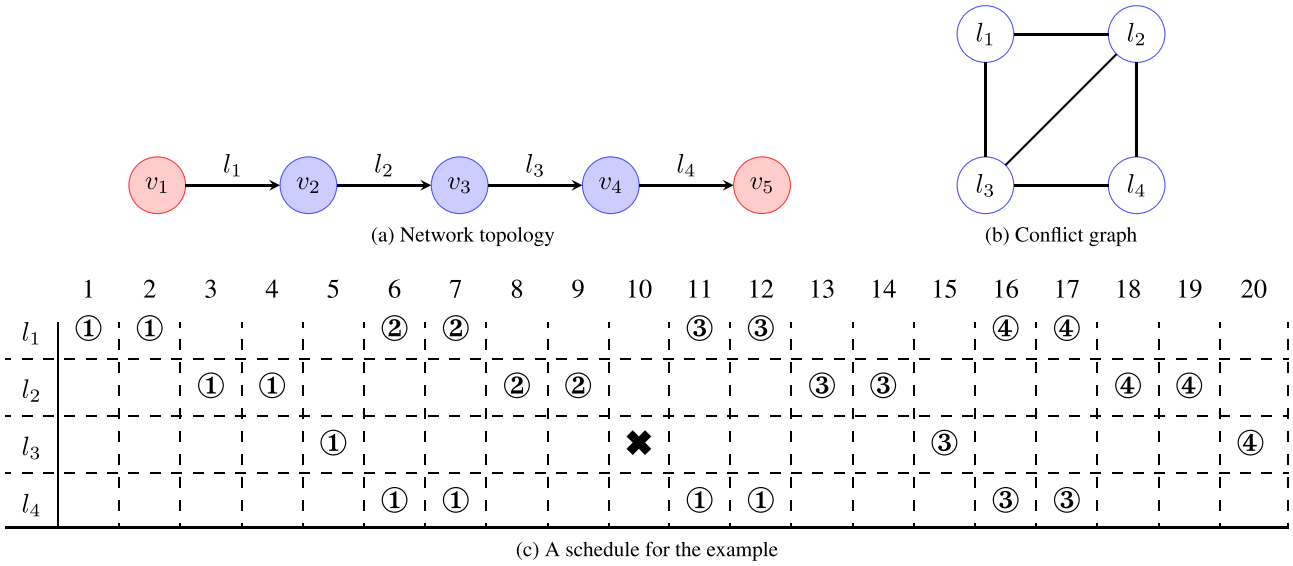


FIGURE 1. Example of a wireless network with 5 nodes and 4 logical links. Notation ① ($i \geq 1$) represents that a link serves recoded packets for the i-th batch, and ✖ depicts that all packets served at the time slot do not arrive at the next hop.

By following these rules, we now illustrate how a periodic scheduling is applied on a toy example. Figures 1a and 1b describe a wireless network and its conflict graph, respectively. In the conflict graph, each vertex represents a link, and two vertices are connected by an edge if the two corresponding links cannot transmit at the same time. The maximal independent set are thus identified as $e_1 = \{l_1, l_4\}$, $e_2 = \{l_2\}$ and $e_3 = \{l_3\}$. Consider $c_1 = 2, c_2 = 2, c_3 = 4, c_4 = 3$, and $t_1 = t_2 = t_3 = 4, t_4 = 6$. Sequentially, we construct a scheduling sequence $\hat{\Upsilon}_{\mathcal{S}_{max}} = e_1, e_1, e_2, e_2, e_3$ in such a way that constraint (5) is held. Figure 1c then depicts a periodic scheduling generated by sequence $\hat{\Upsilon}_{\mathcal{S}_{max}}$. As it can be seen, one batch is delivered to the destination node within every period except for the first one. Note that the second batch is lost at the 10-th slot, then node v_4 has to recode and transmit the most recently arrived batch again as the requirement of Rule 4.

The above example reveals that the scheduling of batches can be generated by a finite sequence $\hat{\Upsilon}_{\mathcal{S}_{max}}$. With the above observation, the objective of the scheduling is now to maximize the network utilization within $\hat{\Upsilon}_{\mathcal{S}_{max}}$. For this purpose, the following optimization problem is derived from (P).

$$
\begin{aligned}
&\underset{\substack{t_1,\cdots,t_n,m\\ \beta_e, e\in\mathcal{S}_{max}}}{\text{maximize}} \frac{mf(t_1,\ldots,t_n)}{\sum_e \beta_e}\\
&\text{subject to } t_k \in \mathbb{N}, \quad t_k > M,\ k = 1,\ldots,l,\\
&\qquad m \in \mathbb{N}, \quad m \leq U,\\
&\qquad \sum_{e\in\mathcal{S}_{max}} \beta_e r_{e,i} \geq m t_i \prod_{j=1}^{i-1}(1-\epsilon_j^{t_j})\\
&\qquad \beta_e \in \mathbb{N}\cup\{0\}, \quad e \in \mathcal{S}_{max},\ \mathbf{r}_e \in \mathcal{R}_{max}. \qquad \text{(PI)}
\end{aligned}
$$

where β_e is defined as the number of independent set e contained in one circle, m denotes the number of batches served in a single scheduling period, and U is the maximum number of batches allowed to be served in a period.

Proposition 2: The optimal value of (P)*is an upper bound on of problem* (PI). *Moreover, problem* (PI) *is equivalent to* (P) *if U is infinite.*

Proof: Let $\beta_e^*, e \in \mathcal{S}_{max}, m^*$ and $t_i^*, i = 1,\ldots,n$, be an optimal solution of problem (PI). Define $T^{*\prime} = \sum_e \beta_e^*/m^*$ and $\alpha_e^{*\prime} = \beta_e^*/\sum_e \beta_e^*$. Suppose $\alpha_e^*, e \in \mathcal{S}_{max}$ and T^* are then optimal solution of problem (P) given the same network settings. The proposition then is an immediate consequence of the fact that $T^{*\prime}$ and $\alpha_e^{*\prime}$ approach T^* and α_e^* as U approaches infinity. □

After solving (PI), we obtain a set of independent sets $e^* \in \mathcal{S}_{max}$ corresponding to non-zero β_e. Then the following proposition provides a criterion to organize the independent sets e^* and shows that any sequence consisting of e^* is a sufficient condition for network stability under our model and proposed scheduling rule.

Proposition 3: Given β_e^*, $e \in \mathcal{S}_{max}$, m^* and t_i^*, $i = 1,\ldots,n$, a periodic scheduling generated by any finite sequence $\Upsilon'_{\mathcal{S}_{max}}$ consisting of β_e^* independent set e is stable and achieves the optimal value for problem (PI) *under the joint recoding and scheduling rules.*

Proof: For simplifying discussion, we assume that every node can receive at least one packet for any batch forwarded by the previous hop. Since the rank for each batch is i.i.d, the optimality of the scheduling is that node v_n can send $m^* \cdot t_n^*$ packets in each circle. To prove it, we first claim that v_n can send $m^* \cdot t_n^*$ packets in each circle as long as node v_{n-1} sends $m^* \cdot t_{n-1}^*$ packets in each circle.

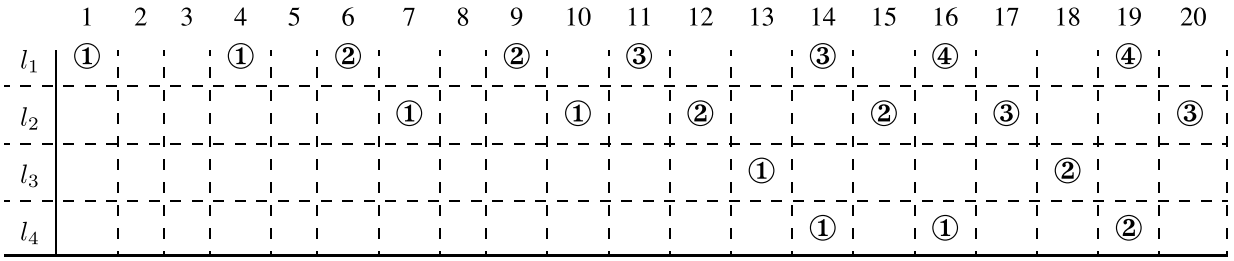


FIGURE 2. Another schedule for the network in Fig. 1.

Consider two consecutive periods P_1 and P_2. Suppose node v_{n-1} sends m^* batches in P_1 and P_2, respectively, whereby v_n sends $m'(< m^*)$ batches in P_1.[3] Without of generality, let $m' = m^* - 1$. Since v_n only sends $m_* - 1$ of m^* received batches in P_1, it must store one batch when the same turn comes in P_2. That is, $m^* \cdot t_n^*$ packets will be delivered in P_2. Due to the periodicity of the scheduling, it can be ensured that $m^* \cdot t_n^*$ packets will be delivered in every subsequent circle, if v_{n-1} sends $m^* \cdot t_{n-1}^*$ packets in these circles.

The same discussion can be directly applied on nodes v_{i+1} and v_i, $i = n-1, \ldots, 1$. Since v_1 is source, the optimality of the scheduling is obtained by induction. The stability of the scheduling then is an immediate consequence of its optimality. Finally due to the definition of the expected rank, it is easily shown that this proof is not compromised by releasing the assumption of the "error-free" transmission of batches. □

Define the delay of a batch as the number of time slots between the first packet for the batch is sent at v_1 and the last packet is sent at v_n. Let $T_p = \sum_{e\in\mathcal{S}_{max}} \beta_e^*$ be the length of $\hat{\Upsilon}_{\mathcal{S}_{max}}$.

Corollary 1: The delay of a batch is smaller than nT_p.

Though Proposition 3 indicates that the order of the independent sets in $\hat{\Upsilon}'_{\mathcal{S}_{max}}$ does not affect the network utilization, it does influence the delay of batches. For instance, consider the network in Fig. 1, again. Figure 2 shows a scheduling constructed by sequence $\hat{\Upsilon}'_{\mathcal{S}_{max}} = e_1, e_2, e_3, e_2, e_1$. Clearly, the delay incurred by $\hat{\Upsilon}'_{\mathcal{S}_{max}}$ is greater than that by $\hat{\Upsilon}_{\mathcal{S}_{max}}$. Figure 2 is an example showing that there exists a scheduling sequence that provides the same throughput performance as that in Fig. 1, but has a longer end-to-end delay even though both sequences have the same independent sets. In particular, the example implies that there always exists a scheduling sequence with a delay greater than $(n-1)\,T_p$. Accordingly, it is necessary to elaborately construct $\hat{\Upsilon}_{\mathcal{S}_{max}}$. However, to find a sequence $\hat{\Upsilon}_{\mathcal{S}_{max}}$, given β_e^*, with minimal delay is an NP-Hard problem in usual. In the next subsection, we will give a greedy algorithm for aligning the independent sets in order to reduce the end-to-end delay.

Finally, the expected rank $\bar{h}_{n+1}$ can be computed in a following matrix fashion [20] by means of the proposed joint rules.

$$f(t_1, \ldots, t_n) = \bar{h}_{n+1} = \mathbf{h}_1 \mathbf{Q} \prod_{i=1}^{n} \Lambda_i \mathbf{Q}^{-1} \mathbf{e}, \tag{8}$$

where Λ_i is an $(M+1)\times(M+1)$ diagonal matrix with eigenvalues

$$\lambda_{i,j} = \begin{cases} 1 & j = 1, \\ \sum_{n=j-1}^{t_i} \binom{t_i}{n} (1-\epsilon_i)^n \epsilon_i^{t_i-n} \zeta_{j-1}^n & j = 2, 3, \ldots, M+1, \end{cases}$$

$$\zeta_r^n = \begin{cases} \prod_{i=0}^{r-1} (1 - q^{-n+i}) & r > 0, \\ 1 & r = 0, \end{cases} \tag{9}$$

and $\mathbf{Q} = [b_{i,j}]_{1\le i,j\le M+1}$ is an $(M+1)\times(M+1)$ lower-triangular matrix with entries

$$b_{i,j} = \begin{cases} 0 & i < j, \\ 1 & j = 1, \\ \zeta_{j-1}^{i-1} & \text{otherwise}, \end{cases}$$

and $\mathbf{e} = [0, 1, \ldots, M]$ and $\mathbf{h}_1 = [0, 0, \ldots, 0, 1]$. Using (8), the complexity of calculating $\bar{h}_{n+1}$ is dominated by the batch size M.

Remark 1: To achieve the upper bound, it does not necessarily set U to infinite. Consider a line network with three nodes. In such a network, only one link can transmit simultaneously, i.e., every independent set contains only one element. If the link capacities are 1, then the optimal solutions for both (P) and (PI) are the same no mater what U is set to.

Remark 2: In this subsection, we show that nodes can recode and transmit batches in a periodic fashion. This periodic scheme has small coding and scheduling overhead and is easy to be implemented. Note that Adaptive Recoding [14], [15] may be able to further improve the inner code performance. However, periodic scheduling cannot be applied to Adaptive Recoding, because many slots will be wasted due to random recoding sizes of batches. In addition, it will involve more integer variables in the design of recoding and scheduling, since the recoding size of a batch depends not only on network condition, but also on the rank of transfer matrix of batches. Thus, the use of Adaptive Recoding will bring both high coding and high scheduling overhead. How to

[3] We say node v_i sends a batch, if v_i transmits t_i recoded packets for the batch.

balance the overhead and the performance for combining Adaptive Recoding and scheduling is an interesting problem but out of the scope of this paper.

Remark 3: Using the periodic scheduling, every batches arriving at the destination node experiences the same delay. Given a sequence Υ, let T_p and d be the sequence size and the corresponding delay of a batch, respectively. Suppose that the destination node successfully decodes an information after N circles. Then the total latency is $(N-1)T_p + d - \varepsilon$, where $\varepsilon > 0$ if $m > 1$ and the decoding is ended before the last circle is completed.

C. APPROXIMATION ALGORITHM

Problem (PI) is a *nonlinear integer programming* (NLIP) problem, which is usually NP-hard [28]. In solving NLIP, One of the main difficulties lies in the combinatorial characteristics of the integer variables. A common method of handling this problem is to find a continuous relaxation of the NLIP, i.e., let β_e $(e \in \mathcal{S}_{max})$, m and t_i $(i = 1, \ldots, n)$ be real numbers. However, these relaxations do not work well for (PI). Due to link capacity $c_i \geq 1$, converting β_e into integers may cause a large deviation from the desired values. Moreover, inequality (5) may not be held after integrating β_e and t_i.

Another difficulty in solving (PI) is that it suffers from the curse of dimensionality. First we notice that the feasible region exponentially depends on U. Corollary 1 also implies that a large U may incur long delay as well as high scheduling overhead. Thus, the value of U should be chosen wisely. However, it is difficult to determine U because it is implicitly related to the network topology, the path and channel conditions as discussed in Remark 1. To facilitate the problem, we thus focus on delivering a batch per period. The optimization problem is then expressed as follows.

$$\begin{aligned}
&\underset{\substack{t_1,\cdots,t_n \\ \beta_e, e\in\mathcal{S}_{max}}}{\text{maximize}} \quad \frac{f(t_1,\ldots,t_n)}{\sum_e \beta_e} \\
&\text{subject to } t_i \in \mathbb{N}, \quad t_i \geq M, \quad i = 1,\ldots,n, \\
&\qquad \sum_e \beta_e r_{e,i} \geq t_i \prod_{j=1}^{i-1}(1-\epsilon_j^{t_j}), \\
&\qquad \beta_e \in \mathbb{N}\cup\{0\}, \quad e \in \mathcal{S}_{max}, \ \mathbf{r}_e \in \mathcal{R}. \qquad \text{(PA)}
\end{aligned}$$

Second, it can be seen that the search space of (PA) still increases exponentially with the length of the sequence $\Upsilon_{\mathcal{S}_{max}}$. To overcome this problem, it is desirable to devise an iterative approach with appropriate step length. Inspired by gradient descent method, we then propose an greedy algorithm that seeks an independent set providing the largest improvement of network utilization at each iteration. Algorithm 1 describes the searching procedure, where $\mathbf{1}$ is an all-one vector with n entries and δ is the stopping criterion.

At the beginning of Algorithm 1, vector $\mathbf{t}$ starts at $\mathbf{1}$ as function f outputs zero with a zero variable. In the while loop, every rate vector $\mathbf{r} \in \mathcal{R}$ is examined in term of its benefit to network utilization. The best solution found is saved and returned as the result, see Step 3. If there exits many solutions, then one of them is selected randomly. In this way, the network utilization is improved along the direction of steepest ascent. Note that, Step 3 can be implemented in parallel. The algorithm keeps track of the best solution found so far and returns it as the final result whenever the stopping criterion is met. Finally, the recoding size used at node v_i is $t_i = \sum_e \beta_e r_{e,i}$.

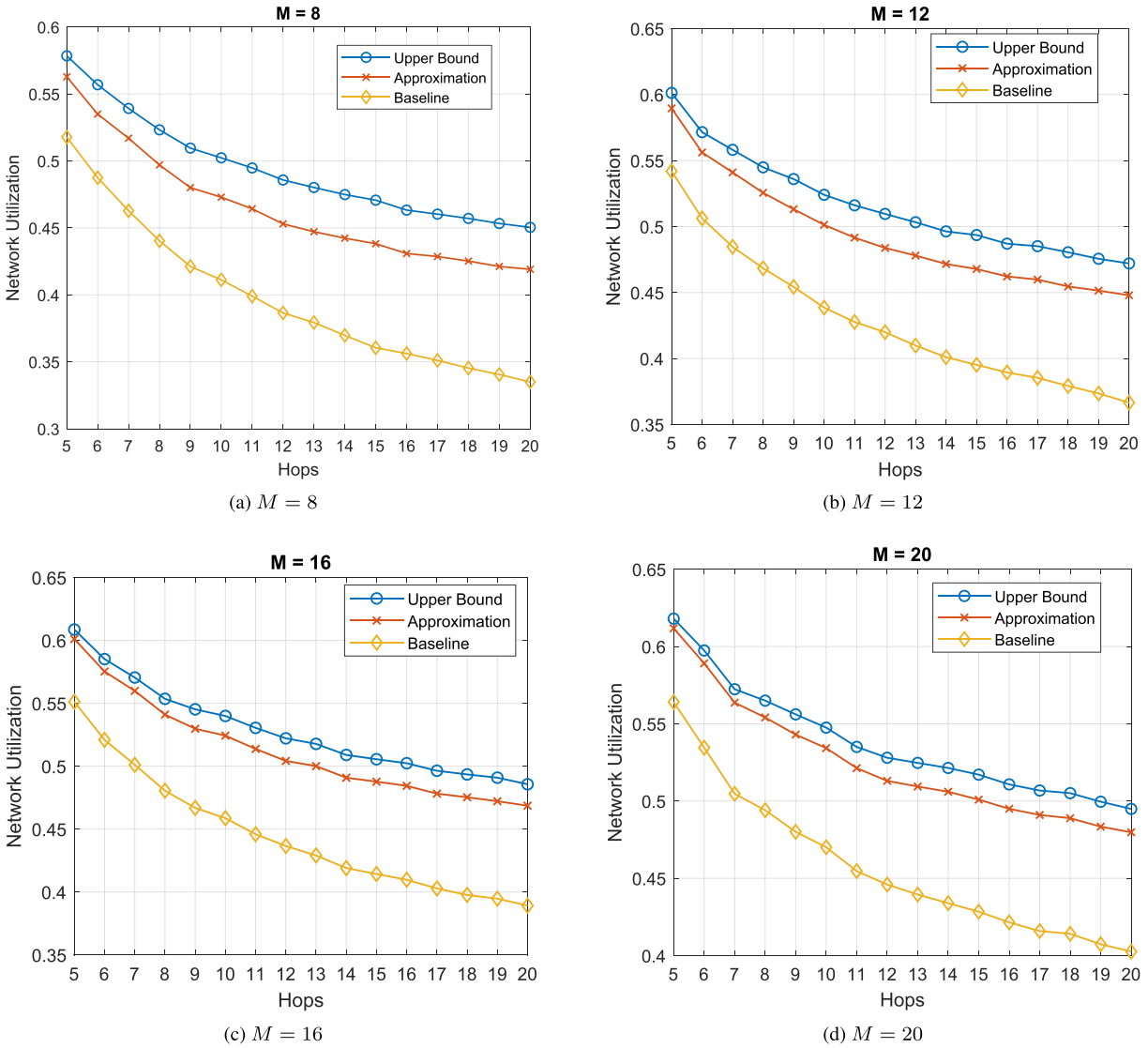
In each iteration, the search space is fixed to $\mathcal{S}_{max}$. Note that function f is non-decreasing and bounded by M. It follows that Algorithm 1 does always converge, and the number of iteration is $\mathcal{O}(M)$. Moreover, equation (8) shows that the operation of function f is matrix multiplication related to batch size M. In summary, the computation complexity of Algorithm 1 is $\mathcal{O}(M^4\mathcal{S}_{max})$. To further reduce the complexity,

Algorithm 1 Solve Problem (PA)

Input: $M, n, \epsilon_1, \epsilon_2, \ldots, \epsilon_l, \mathcal{R}$;
Output: $\mathbf{t}, \beta_e^*(e \in \mathcal{S}_{max})$;
1: Initialize $\mathbf{t} \leftarrow \mathbf{1}$, $\eta^* = 0$ and set all β_e^*'s to 0;
2: **while** True **do**
3: Find a rate vector $\mathbf{r}^* = \arg\max_{\mathbf{r}\in\mathcal{R}} \frac{f(\mathbf{t}+\mathbf{r})}{(\sum_e \beta_e^*+1)}$;
4: Calculate network utilization $\eta' = \frac{f(\mathbf{t}+\mathbf{r}^*)}{(\sum_e \beta_e^*+1)}$;
5: **if** $\eta' - \eta^* \geq \delta$ or the minimum entry of $\mathbf{t}$ is less then M **then**
6: $\eta^* \leftarrow \eta'$, $\mathbf{t} \leftarrow \mathbf{t} + \mathbf{r}^*$ and $\beta_{r^*}^* = \beta_{r^*}^* + 1$;
7: **else**
8: **return** $\mathbf{t}, \beta_e^*(e \in \mathcal{S}_{max})$;
9: **end if**
10: **end while**

Algorithm 2 Construct a Scheduling Sequence $\hat{\Upsilon}_{\mathcal{S}_{max}}$.

Input: $\alpha_e^*, \mathcal{S}_{max}$;
Output: $\Upsilon_{\mathcal{S}}$;
1: Construct set $\mathcal{S}_{max}^* = \{e : e \in \mathcal{S}_{max}, \alpha_e^* \neq 0\}$;
2: **for** $i = 1$ to n **do**
3: $\mathcal{D}_i \leftarrow \varnothing$;
4: **for** $j = 1$ to $|\mathcal{S}_{max}^*|$ **do**
5: **if** l_i belongs to $e_j (\in \mathcal{S}_{max}^*)$ **then**
6: $\mathcal{D}_i \leftarrow \mathcal{D}_i \bigcup \{e_j\}$;
7: $\mathcal{S}_{max}^* \leftarrow \mathcal{S}_{max}^* - \{e_j\}$;
8: **end if**
9: **end for**
10: **end for**
11: **for** $i = 1$ to n **do**
12: **for** $j = 1$ to $|\mathcal{D}_i|$ **do**
13: Select $e \in \mathcal{D}_i$ randomly;
14: $\mathcal{D}_i \leftarrow \mathcal{D}_i - \{e_j\}$;
15: Insert $\langle \alpha_e^*, e \rangle$ into the tail of sequence $\Upsilon_{\mathcal{S}}$
16: **end for**
17: **end for**



(a) $M = 8$ (b) $M = 12$

(c) $M = 16$ (d) $M = 20$

FIGURE 3. Network utilization: the proposed methods *vs* upper bound.

it can use the following formula to approximate f.

$$\tilde{f} \triangleq \lim_{q\to\infty} f = \sum_{r=1}^{M}\prod_{k=1}^{l} I_{1-\epsilon_i}(r, t_k - r + 1). \tag{10}$$

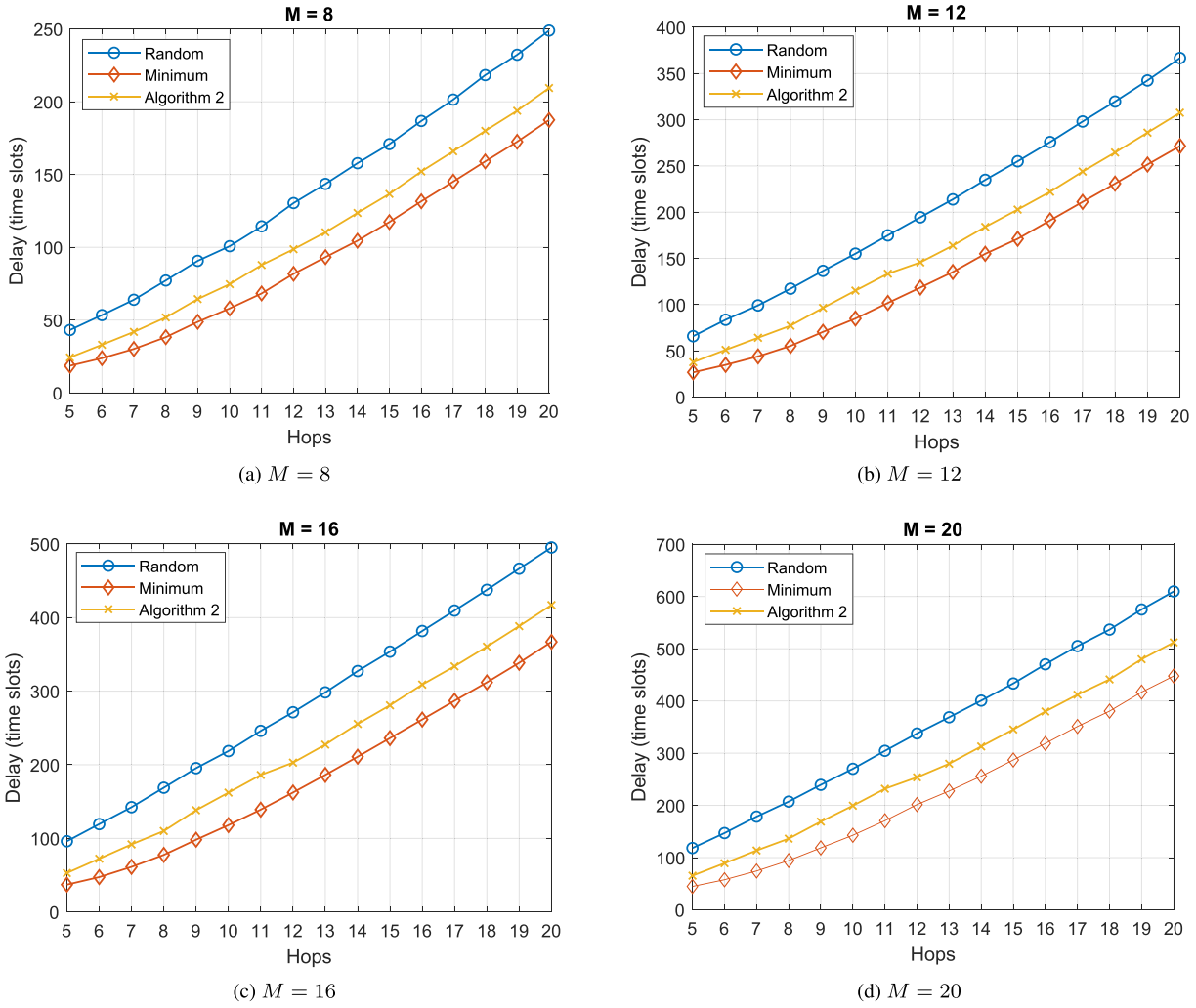
where $I_{1-\epsilon_j}(r, t_j - r + 1)$ is a regularized incomplete beta function. By using (10) the computation complexity of Algorithm 1 is $\mathcal{O}(M^2\mathcal{S}_{max})$.

On the other hand, in order to construct a scheduling sequence with small end-to-end delay, we design the following greedy algorithm. Let $\beta_e^*, e \in \mathcal{S}_{max}$, are the optimal solution of (PI). First, we construct a set $\mathcal{S}_{max}^*$ consisting of the independent sets $e \in \mathcal{S}_{max}$ corresponding to non-zero β_e^*. Next, we pick out all independent sets including link l_1 from $\mathcal{S}_{max}^*$, forming a new set $\mathcal{D}_1 = \{e : l_1 \in e, e \in \mathcal{S}_{max}^*\}$. In a same way, we find sets $\mathcal{D}_i = \{e : l_i \in e, e \in \mathcal{S}_{max}^*, e \notin \mathcal{D}_j, j = 1, \ldots, i-1\}, i = 2, \ldots, n$. Note that, set $\mathcal{D}_i (i > 1)$ can be an empty set. Finally, a two-tuple $\langle \beta_e^*, e\rangle (e \in \mathcal{D}_i)$ is placed at the tail of sequence $\hat{\Upsilon}_{\mathcal{S}_{max}}$ if it satisfies: 1. All independent sets belonging to sets $\mathcal{D}_j, j = 1, \ldots, i-1$, have been already inserted into $\hat{\Upsilon}_{\mathcal{S}_{max}}$; 2. Set e has not been chosen yet. The pseudocode is given in Algorithm 2. The complexity of Algorithm 2 is $\mathcal{O}(n\mathcal{S}_{max})$.

After starting a transmission procedure, $\hat{\Upsilon}_{\mathcal{S}_{max}}$ will be repeatedly ran until all batches are delivered to the destination node. Note that Algorithm 2 is suitable for both problems (PI) and (PA).

V. NUMERICAL EVALUATIONS

The experiments are conducted in MATLAB, and all optimization problems are solved by means of the optimization toolbox with default settings. In the last section, we proposed two algorithms to solve (PA) and combine the independent sets, respectively. In this section, we then numerically evaluate the performances of these algorithms.



(a) $M = 8$ (b) $M = 12$ (c) $M = 16$ (d) $M = 20$

FIGURE 4. **Delay: the proposed methods *vs* minimum delay.**

In the evaluation, link networks, e.g. Fig. 1a, of length l ranges from 5 to 20 are tested. Both primary and secondary conflicts are considered. That is, nodes within two hops cannot transmit simultaneously. As a result, links l_i, $l_{i\pm1}$ and $l_{i\pm2}$ cannot be in the same independent sets. Network links are heterogeneous with respect to packet loss rate and link capacity. More specific, the packet loss rates ϵ_i and the link capacities c_i on all the links are independently and uniformly chosen over the interval [0.05, 0.35] and [2, 4], respectively, and remain constant during the evaluations. The field size is $q = 2^8$. The BATS codes have four different batch sizes 8, 12, 16 and 20. For each combination of l and M (e.g $l = 5$, $M = 12$), the following experiment is repeated 1000 times.

We first evaluate the network utilization. Fig. 3 illustrates the network utilization of the following approaches.

- Upper Bound is obtained by solving (P).
- Approximation is the optimal values of (PA).
- Baseline is obtained by solving (P) with fixed $t_i = M$ for $i = 1, 2, \ldots, l$.

We compare our proposed algorithms with the upper bound, since this bound tells up the potential gain in terms of the network utilization using the inner code designs proposed in this paper. We observe that all the decreases are less than 8 percent compared to Upper Bound. In particular, the gap between the upper bound and our proposed algorithms tends to be smaller when M increases. On the other hand, Baseline represents the maximum network utilization can be achieved by using the inner code in [8]. Fig. 3 shows that our proposed algorithm improves significantly the performance compared with Baseline. Moreover, the performance of Baseline degrades faster than of Approximation. This indicates that the recoding size must adapt to channel conditions such as packet loss rate.

The average end-to-end delay of a batch occurred by the following approaches is shown in Fig. 4. The independent sets are the solutions of (PA).

- Random is to organize the independent sets randomly.
- Algorithm is to order the set by means of Algorithm 2.

- Minimum is the minimum delay that is found by a brute-force search.

It is easily seen that there is a huge gap between Random and Minimum even if the value of m is set to 1 in problem (PA). Therefore, it is necessary to find a way of ordering the independent sets to reduce the end-to-end delay. Algorithm 2 then provides a simple but efficient solution.

Combining Fig. 3 with Fig. 4, we see that the network utilization is in conflict with the end-to-end delay in term of batch size. Larger batch size implies a better utilization, but inevitably increases the delay. Moreover, the complexity of decoding is proportional to the batch size. As a result, the batch size should be carefully designed so as to match the system requirements. We will discuss this problem in the future work.

VI. CONCLUSION

In this work, we discussed the joint design of the inner code and scheduling, taking the network utilization into account. By analyzing the relationship of the network utilization to both inner codes and scheduling, we formulated a joint recoding-scheduling optimization problem and showed that a scheduling built by a finite length sequence can provide the near-optimal performance. To efficiently solve the problem, we then constructed an approximation algorithm searching a solution in a limited space. We further proposed the ordering algorithm that combines the independent sets in an ascending order with respect to the distance from nodes to the source, and thus can reduce the scheduling delay. The simulation results show the efficiency of our algorithms in terms of the network utilization and the end-to-end delay.

APPENDIX

PROOF OF PROPOSITION 1

Proof: It is easily seen from (4) that the expected rank of transfer matrices is determined by both matrices Φ and D. Since the transmissions of each packet at each node are i.i.d, the rank distributions of D's for transfer matrices are the same. Hence, we focus on the rank distribution of Φ in the following discussion.

Let $\Phi_i^{(b)}$ and $\Phi_i^{(a)}$ be the recoding matrices corresponding to the recoding policies that node v_i starts recoding a batch before and after v_{i-1} stops transmitting the batch, respectively. Denoted by $\mathrm{rk}(\Phi_i)$ the rank of Φ_i. Then the inequality $\Pr\{\mathrm{rk}(\Phi_i^{(a)}) \geq r\} \leq \Pr\{\mathrm{rk}(\Phi_i^{(b)}) \geq r\}$, $0 \leq r \leq M$ is an immediate consequence of the fact that the space spanned by $\Phi_i^{(b)}$ is no greater than that by $\Phi_i^{(a)}$. Moreover, the equality holds only if v_i receives no more packets for a batch over the transmission of the batch.

Next, we claim that $\Pr\{\mathrm{rk}(\mathbf{H}_{n+1}^{(b)}) \geq r\} \leq \Pr\{\mathrm{rk}(\mathbf{H}_{n+1}^{(a)}) \geq r\}$, $r > 0$, where $\mathbf{H}_{n+1}^{(b)} = \mathbf{H}_n^{(b)}\Phi_n^{(b)}\mathbf{D}_n$ and $\mathbf{H}_{n+1}^{(a)} = \mathbf{H}_n^{(a)}\Phi_n^{(a)}\mathbf{D}_n$. This can be proved by induction on n, the number of hops. For $n = 1$, we have

$$\begin{aligned}\Pr\{\mathrm{rk}(\mathbf{H}_2^{(b)}) \geq r\} &= \Pr\{\mathrm{rk}(\Phi_1^{(b)}\mathbf{D}_1) \geq r\} \\ &\leq \Pr\{\mathrm{rk}(\Phi_1^{(a)}\mathbf{D}_1) \geq r\} = \Pr\{\mathrm{rk}(\mathbf{H}_2^{(a)}) \geq r\}.\end{aligned}$$

Suppose that the claim holds for $n = l - 1$. Then, consider $n = l$. We write

$$\begin{aligned}\Pr\{\mathrm{rk}(\mathbf{H}_{l+1}^{(b)}) \geq r\} &= \Pr\{\mathrm{rk}(\mathbf{H}_l^{(b)}\Phi_l^{(b)}\mathbf{D}_l) \geq r\} \\ &\leq \Pr\{\mathrm{rk}(\mathbf{H}_l^{(a)}\Phi_l^{(b)}\mathbf{D}_l) \geq r\} \\ &\leq \Pr\{\mathrm{rk}(\mathbf{H}_l^{(a)}\Phi_l^{(a)}\mathbf{D}_l) \geq r\} \\ &= \Pr\{\mathrm{rk}(\mathbf{H}_{l+1}^{(a)}) \geq r\}.\end{aligned}$$

Therefore, the claim is proved by induction. With this claim, we can obtain

$$\begin{aligned}\bar{h}_{n+1}^{(a)} &= \sum_{j=0}^{M} j h_{n+1,j} = \sum_{j=1}^{M} \Pr\{\mathbf{H}_{n+1}^{(a)} \geq j\} \\ &\leq \sum_{j=1}^{M} \Pr\{\mathbf{H}_{n+1}^{(b)} \geq j\} = \bar{h}_{n+1}^{(b)}.\end{aligned}$$

The equality holds only if nodes v_i, $i = 1, \ldots, n$, receive no more packets for a batch within the recoding process of the batch. The proof of the theorem is completed.

□

REFERENCES

[1] M. S. Khan, D. Midi, M. I. Khan, and E. Bertino, "Fine-grained analysis of packet loss in MANETs," *IEEE Access*, vol. 5, pp. 7798–7807, 2017.
[2] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP performance," *IEEE Trans. Mobile Comput.*, vol. 4, no. 2, pp. 209–221, Mar. 2005.
[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1206, Jul. 2000.
[4] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.
[5] D. Li, X. Guang, Z. Zhou, C. Li, and C. W. Tan, "Hierarchical performance analysis on random linear network coding," *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 2009–2021, May 2018, doi: 10.1109/tcomm.2017.2787991.
[6] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[7] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, *RaptorQ Forward Error Correction Scheme for Object Delivery*, document TS RFC6330, Internet Engineering Task Force, 2011. [Online]. Available: http//:tools.ietf.org/html/rfc6330
[8] S. Yang and R. W. Yeung, "Batched sparse codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5322–5346, Sep. 2014.
[9] S. Yang, and R. W. Yeung, *BATS Codes: Theory and Practice* (Synthesis Lectures On Communication Networks). San Rafael, CA, USA: Morgan & Claypool, 2017.
[10] Q. Huang, K. Sun, X. Li, and D. Wu, "Just FUN: A joint fountain coding and network coding approach to loss-tolerant information spreading," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Philadelphia, PA, USA, Aug. 2014, pp. 83–92.
[11] H. Zhang, K. Sun, Q. Huang, Y. Wen, and D. Wu, "FUN coding: Design and analysis," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3340–3353, Dec. 2016.
[12] N. Sprea, M. Bashir, D. Truhachev, K. V. Srinivas, C. Schlegel, and C. Sacchi, "BATS coding for underwater acoustic communication networks," in *Proc. OCEANS*, Marseille, France, Jun. 2019, pp. 1–10.
[13] J. Yue, M. Xiao, and Z. Pang, "Distributed BATS-based schemes for uplink of industrial Internet of Things," in *Proc. ICC*, Shanghai, China, May 2019, pp. 1–6.

[14] B. Tang, S. Yang, B. Ye, S. Guo, and S. Lu, "Near-optimal one-sided scheduling for coded segmented network coding," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 929–939, Mar. 2016.
[15] H. H. Yin, S. Yang, Q. Zhou, and L. M. Yung, "Adaptive recoding for BATS codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 2349–2353.
[16] X. Xu, M. S. G. Praveen Kumar, Y. L. Guan, and P. H. Joo Chong, "Two-Phase Cooperative Broadcasting Based on Batched Network Code," *IEEE Trans. Commun.*, vol. 64, no. 2, pp. 706–714, Feb. 2016.
[17] Y. Gao, X. Xu, Y. L. Guan, and P. H. J. Chong, "V2X content distribution based on batched network coding with distributed scheduling," *IEEE Access*, vol. 6, pp. 59449–59461, 2018.
[18] S. Yang, T.-C. Ng, and R. W. Yeung, "Finite-length analysis of BATS codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 322–348, Jan. 2018.
[19] J. Yang, Z.-P. Shi, C.-X. Wang, and J.-B. Ji, "Design of optimized sliding-window BATS codes," *IEEE Commun. Lett.*, vol. 23, no. 3, pp. 410–413, Mar. 2019.
[20] Z. Zhou, C. Li, S. Yang, and X. Guang, "Practical inner codes for BATS codes in multi-hop wireless networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2751–2762, Mar. 2019.
[21] H. H. F. Yin, X. Xu, K. H. Ng, Y. Liang Guan, and R. W. Yeung, "Packet efficiency of BATS coding on wireless relay network with overhearing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 1967–1971.
[22] C. Joo, X. Lin, and N. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1132–1145, Aug. 2009.
[23] S. Yang, S.-W. Ho, J. Meng, and E.-H. Yang, "Capacity analysis of linear operator channels over finite fields," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4880–4901, Aug. 2014.
[24] C. Perkins, E. Belding-Royer, and S. Das, *Ad Hoc on-Demand Distance Vector (AODV) Routing*, document IETF RFC 3561, Jul. 2003.
[25] Y. Gao, X. Xu, Y. Zeng, and Y. L. Guan, "Multi-hop video streaming with network coding in vehicular networks," in *Proc. Veh. Technol. Conf.*, Nov. 2018, pp. 1–5.
[26] G. Chen, X. Cao, L. Liu, C. Sun, and Y. Cheng, "Joint scheduling and channel allocation for end-to-end delay minimization in industrial wirelessHART networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2829–2842, Apr. 2019.
[27] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, Jan. 2010.
[28] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surv. Oper. Res. Manage. Sci.*, vol. 17, no. 2, pp. 97–106, Jul. 2012.
[29] A. Dana, R. Gowaikar, R. Palanki, B. Hassibi, and M. Effros, "Capacity of wireless erasure networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 789–804, Mar. 2006.
[30] M. Zelen and N. C. Severo, *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*, M. Abramowitz and I. A. Stegun, Eds. New York, NY, USA: Dover, 1972.

ZHIHENG ZHOU (Member, IEEE) received the B.S. and Ph.D. degrees from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2007 and 2014, respectively. From 2015 to 2017, he was a Postdoctoral Research Fellow with the Institute of Network Coding, The Chinese University of Hong Kong. He is currently a Lecturer with UESTC. His major research interests include network coding, reliability study, and network optimization.

JINGMIN KANG is currently pursuing the M.E. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China. His major research interests include network coding, the Internet of Things, and network optimization.

LIANG ZHOU is currently a Professor with the University of Electronic Science and Technology of China, China. His research interests include wireless communications and networking, error control coding, information system engineering, and communication and cyberspace security.