Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 390457, 9 pages
http://dx.doi.org/10.1155/2015/390457

Research Article

Minimum-Energy Wireless Real-Time Multicast by Joint Network Coding and Scheduling Optimization

Guoping Tan,[1] Yandan Zhu,[1] Yueheng Li,[1] Lili Zhang,[1] and Jing Xu[2]

[1]*College of Computer and Information Engineering, Hohai University, Nanjing 210098, China*
[2]*Shanghai Microsystem and Information Technology Research Institute, Chinese Academy of Sciences, Shanghai 200050, China*

Correspondence should be addressed to Guoping Tan; gptan@hhu.edu.cn

Received 21 January 2015; Revised 31 March 2015; Accepted 31 March 2015

Academic Editor: Joaquim Joao Judice

For real-time multicast services over wireless multihop networks, to minimize the energy of transmissions with satisfying the requirements of a fixed data rate and high reliabilities, we construct a conflict graph based framework by joint optimizing network coding and scheduling. Then, we propose a primal-dual subgradient optimization algorithm by random sampling K maximal stable sets in a given conflict graph. This method transforms the NP-hard scheduling subproblem into a normal linear programming problem to obtain an approximate solution. The proposed algorithm only needs to adopt centralized technique for solving the linear programming problem while all of the other computations can be distributed. The simulation results show that, comparing with the existing algorithm, this algorithm can not only achieve about 20% performance gain, but also have better performance in terms of convergence and robustness.

1. Introduction

Since Ahlswede et al. proposed network coding (NC) [1], many studies have shown that NC can not only increase the throughput significantly but also achieve better robustness. Actually, it can achieve network multicast capacity by using random network coding (RNC) over a multihop wireless network [2]. This has promoted many studies on RNC based distributed optimization algorithms. For example, Lun et al. proposed to decompose the multicast optimization problem into two subproblems [3]: one is to search for a NC subgraph with minimum cost by modelling as a linear or convex program problem, which can be solved by a distributed primal-dual subgradient optimization algorithm; the other is to design a network coding scheme for the optimal subgraph obtained from the first subproblem; then a simple RNC solution can be employed for this subproblem. Similarly, Wu et al. proposed a distributed NC optimization algorithm over mobile ad hoc networks to minimize the energy for multicast services [4]. Lee and Vishwanath recently proposed a distributed algorithm for rate allocations to achieve the network capacity with minimizing the operation cost [5].

Another research direction is to introduce scheduling techniques in medium access control (MAC) layer into an optimization framework. The research in [3, 6] has shown that network performance can be improved significantly by optimizing a scheduling technique in MAC layer. Recently, using interference graph model, Jaramillo et al. have studied an optimization problem on resource scheduling when real-time and non-real-time services coexist in wireless multihop networks [7]. In addition, they have also studied the optimal rate allocation problem under heterogeneous delay constraints [8]. Although these studies cover the resource allocation and scheduling optimization problem for real-time services, they do not introduce NC into the optimization framework.

Apparently, we must integrate the two problems mentioned above together for achieving the best overall performance. By taking both NC and scheduling into account jointly, Rajawat and Giannakis proposed a joint optimization technique to improve the wireless multicast throughput performance under strict delay constraints [9]. Using hyperarcs to model the natural properties of wireless multicasts, Traskov et al. proposed a conflict graph model to

identify effective network settings for studying joint NC and scheduling optimization algorithms [10]. In fact, this conflict graph based framework can be used to build an interference model for those active nodes in wireless networks. In order to avoid interference, we can select an efficient scheduling policy by sampling stable sets in a conflict graph. The studies have shown that, comparing with the scheduling technique with simple orthogonal models, the joint scheduling and NC graph optimization algorithm can improve the multicast throughput significantly [10].

Unfortunately, the joint optimization framework proposed in [10] cannot be used for wireless real-time multicast services without changes, because real-time multicast services usually need networks to support a predefined fixed and qualified data rate. By addressing this issue, Lun et al. [3] and Wu et al. [4] proposed a NC subgraph based optimization framework with minimum cost for supporting fixed multicast rates. However, they do not integrate scheduling techniques into their optimization frameworks. Moreover, since the conflict graph based scheduling optimization problem built in [10] is a NP-hard problem, they proposed a greedy algorithm with sampling the maximum weight stable sets. Nevertheless, there is a major drawback in this algorithm: it is very sensitive to the parameters of iterations such as the step size. In other words, the results often fall far short of the global optimum value in case that the parameters of iterations cannot be set properly. Since those parameters of iterations can only be chosen through trial and error methods, it is thus difficult to meet real-time requirements.

By addressing the problems mentioned above, to minimize the energy with satisfying the requirements of real-time multicast services in wireless multihop networks, we will construct a conflict graph based framework for designing joint NC subgraph and scheduling optimization algorithms. Here we would like to point out that this framework is mainly inspired from [10], but there are two important differences: it focuses on real-time services with a fixed and qualified data rate rather than non-real-time services; the optimization target is to minimize the energy of transmissions rather than to maximize the network capacity. Afterwards, using Lagrangian relaxation, we will propose a joint optimization algorithm by sampling the maximum stable sets randomly in a conflict graph, which includes two steps: first, it randomly samples a certain number of maximum stable sets of the conflict graph at each iteration; then, it solves a scheduling optimization problem by searching for those random sets. This method transforms a NP-hard scheduling subproblem into a plain linear programming problem so that it can be solved efficiently. More importantly, the accuracy can be adjusted not only through the parameters of iterations, but also by the number of random samplings. It thus can have good convergence and robustness.

The remainder of this paper is organized as follows. The network model with a directed hypergraph and its corresponding conflict graph is introduced in Section 2. A conflict graph based real-time multicast optimization framework is presented in Section 3. Using Lagrangian relaxation, a joint optimization algorithm is proposed in Section 4. The numerical simulation results are presented in Section 5. Finally, we conclude the paper in Section 6.

2. Network Model

Considering a wireless multihop network, we use a directed hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ (where $\mathcal{N}$ denotes the set of nodes and $\mathcal{A}$ is a collection of hyperarcs) to represent the model. We define $(i, J) \in \mathcal{A}$ as a hyperarc and $N(i) \subset \mathcal{N}$ as a set of neighbors of node i. When node i sends data, all nodes in $N(i) \subset \mathcal{N}$ are assumed to be within the receiving range. For any hyperarc $(i, J) \in \mathcal{A}$, we have $i \in \mathcal{N}$ and $J \subset \mathcal{N}$. For each node, there are at most $2^{|N(i)|} - 1$ hyperarcs.

The scheduling problem studied here is a scheduling for all of the hyperarcs defined above. When scheduling multiple hyperarcs for transmissions, we must avoid the interference caused by those conflict nodes. The specific hyperarc conflict situation depends on the network interference model. We consider the following two commonly used protocol interference models: the primary interference model and the secondary interference model. It is assumed that each node can only receive data from one node every time in the primary interference model, while in the secondary interference model, besides the above constraints, it is also assumed that any receiving node can only receive data correctly when all other neighbors are in a dormant state. They are strictly defined as follows.

Interference Models. For any two simultaneous scheduling hyperarcs (i_1, J_1) and (i_2, J_2), the necessary and sufficient conditions for no conflict are as follows:

(1) $i_1 \neq i_2$ and $i_1 \notin J_2, i_2 \notin J_1$;

(2) there is $J_1 \cap J_2 = \emptyset$ for the primary interference model; there is $J_1 \cap N(i_2) = \emptyset$ and $J_2 \cap N(i_1) = \emptyset$, where $\emptyset$ is an empty set for the secondary interference model.

Note that in both the primary interference model and the secondary interference model parameters are defined symmetrically. Therefore, we can use an undirected graph to express the conflict among hyperarcs. In this paper, we use the method presented in [10] to formulate an undirected graph based conflict graph. It is defined as follows.

Conflict Graph $\mathcal{G}$. Given a directed hypergraph $\mathcal{H}$ defined above, we can formulate an undirected graph $\mathcal{G} = (\mathcal{T}, \mathcal{B})$, which represents conflicts among all hyperarcs under an interference model. The vertex set $\mathcal{T}$ in $\mathcal{G}$ is the collection of all the hyperarcs in $\mathcal{H}$. Each edge in $\mathcal{B}$ represents the conflict between two connected vertices according to an interference model. That is, the vertices in $\mathcal{G}$ represent the hyperarcs in $\mathcal{H}$; each edge in $\mathcal{G}$ represents a conflict between the two connected vertices. The two corresponding hyperarcs then cannot be scheduled at the same time.

Clearly, based on the above definitions and a specific interference model, we can easily formulate the corresponding conflict graph $\mathcal{G}$ according to any hypergraph $\mathcal{H}$. For a conflict graph $\mathcal{G}$, we now define any subset that there is no edge connected between any two nodes as a stable set S. A

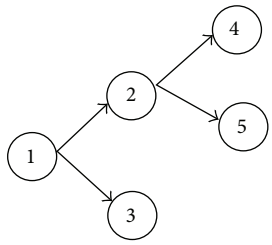

Figure 1: Example of a directed hypergraph $\mathcal{H}$. Here the node set $\mathcal{N} = \{1, 2, 3, 4, 5\}$ and the hyperarc set $\mathcal{A} = \{(1, 2), (1, 3), (1, \{2, 3\}), (2, 4), (2, 5), (2, \{4, 5\})\}$.

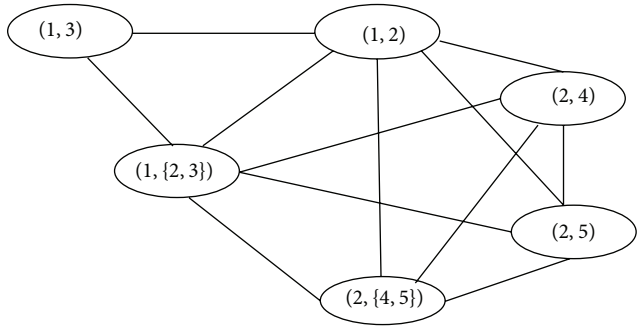

Figure 2: Example of the undirected conflict graph $\mathcal{G}$ corresponding to the hypergraph in Figure 1.

stable set S can be indicated by a column vector of length $|\mathcal{T}|$, which is defined as

$$I_t^S = \begin{cases} 1, & \text{if } t \in S \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

A maximal stable set is one that is not contained in any other stable set. A maximum stable set is a stable set of largest cardinality. The stability numbers of $\mathcal{G}$ are the cardinality of the maximum stable set. The stable set polytope (denoted by $\mathrm{CH_{SS}}$) is the convex hull of the incidence vectors of all stable sets of $\mathcal{G}$. For example, assuming that there are M stable sets in $\mathcal{G}$ and $I_t^{S(i)}$ is the incidence vector of the stable set $S(i)$, then we have

$$\mathrm{CH_{ss}} = \left\{ P \mid P = \sum_{i=1}^{M} \rho_i I_t^{S(i)},\ \forall \rho_i \geq 0,\ \sum_{i=1}^{M} \rho_i = 1 \right\}. \quad (2)$$

Finally, we take an example to illustrate the notations. A directed hypergraph with five nodes is shown in Figure 1 and its corresponding conflict graph in Figure 2.

Note that both the primary and the secondary interference models lead to the same conflict graph. $\mathcal{G}$ has a node for each hyperarc in $\mathcal{H}$; that is, $\mathcal{T} = \mathcal{A} = \{(1, 2), (1, 3), (1, \{2, 3\}), (2, 4), (2, 5), (2, \{4, 5\})\}$. Figure 2 shows that the stable set polytope for this example is the convex hull of the incidence vectors of the five stable sets $\{(1, 3), (2, 4)\}$, $\{(1, 3), (2, 5)\}$, $\{(1, 3), (2, \{4, 5\})\}$, $\{(1, 2)\}$, and $\{(1, \{2, 3\})\}$.

3. Optimization Framework

To integrate NC subgraph optimization problem with scheduling problem together in one framework, we need to define some essential NC variables. First, we define the rate of NC data packet injected to hyperarc (i, J) by node i as z_{iJ}. It is referred to as the coded packet rate in this paper. Then, we use a collection $Z = (z_{iJ})_{(i,J)\in\mathcal{A}}$, also known as a NC subgraph, to represent the coded packet rate injected to all the hyperarcs. For simplifying the optimization framework for supporting real-time multicast services requiring high reliabilities, we assume that the physical layer can ensure perfect reliable transmissions through appropriate power control, channel coding, modulation, and other methods when the coded packet rate on any hyperarc is not more than z_{iJ}. This assumption is realistic for some emerging real-time services such as Tactile Internet, which requires an end-to-end failure rate of 10^{-7} [11]. That is, when the transmission power is large enough within a certain range, this requires guaranteeing a very low error rate for each active link by suitable techniques. It thus leads to a reasonable assumption of perfect reliable links. Finally, we would like to point out that our framework can be extended for matching the real-time services with nonnegligible error rates, for example, a more realistic signal-to-interference-and-noise ratio (SINR) threshold based framework, which needs further studies in the future.

Then, we use the variable $x_{iJj}^{(t)}$ to indicate the transmission rate of the information flow transferred to the terminal $t \in T$ when transmitting from $i \in \mathcal{N}$ to $j \in N(i)$ in case that the coded packet rate injected to the hyperarc $(i, J) \in \mathcal{A}$ is z_{iJ}. Here T represents the collection of all sink nodes in a multicast session. To simplify the notation, define the variable $x_{ij}^{(t)}$ as follows:

$$x_{ij}^{(t)} = \sum_{J \subset N(i)} x_{iJj}^{(t)}. \quad (3)$$

According to (3), the variable $x_{ij}^{(t)}$ can be viewed as the transmission rate of the information flow transferred to the sink node $t \in T$ when transmitting from $i \in \mathcal{N}$ to $j \in N(i)$.

Now let us focus on the optimization target of minimizing the energy of real-time multicast. The objective function can be expressed as a function of the coded packet rate z_{iJ}, which is defined similar to [3] as

$$f_{iJ}(z_{iJ}) = \zeta_{iJ} z_{iJ}. \quad (4)$$

In this function, ζ_{iJ} represents the energy required when the coded packet transferred with the rate z_{iJ} over the hyperarc $(i, J) \in \mathcal{A}$. For real-time multicast services requiring low reliabilities, we can modify this function by introducing some factors for describing transmission costs under different link qualities. However, it is beyond the scope of this paper and left to future studies.

Note that the definition of the function in [3] is only suitable for a single communication link, while the definition used here is for the energy consumption of a hyperarc. Therefore, we assume that the value of ζ_{iJ} depends on the link

with the longest distance over a hyperarc (i, J). Let d_{ij} denote the distance between any two nodes (i, j); then we have

$$\zeta_{iJ} = \max_{j \in J} d_{ij}^2. \tag{5}$$

Using the definitions above, we now can model the optimization problem of minimizing the energy of transmissions with a fixed multicast rate as a joint scheduling and NC subgraph optimization problem; that is,

$$\text{minimize} \quad \sum_{(i,J) \in \mathscr{A}} f_{iJ}(z_{iJ}), \tag{6}$$

s.t.

Capacity constraints are

$$x_{ij}^{(t)} \le \sum_{\{J | j \in J, J \subset N(i)\}} z_{iJ}, \quad \forall i \in \mathcal{N},\ j \in N(i),\ t \in T. \tag{7}$$

Flow constraints C_F are

$$\sum_{j \in N(i)} x_{ij}^{(t)} - \sum_{\{j | i \in N(j)\}} x_{ji}^{(t)} = \begin{cases} R, & i = s \\ -R, & i = t \\ 0, & \text{else} \end{cases} \quad \forall i \in \mathcal{N},\ t \in T \tag{8}$$

$$x_{ij}^{(t)} \ge 0, \quad \forall i \in \mathcal{N},\ j \in N(i),\ t \in T.$$

Scheduling constraints are

$$Z = (z_{iJ})_{i \in \mathcal{N}} \in \mathrm{CH_{SS}}. \tag{9}$$

Note that the objective function in this optimization framework is a convex function and $\mathrm{CH_{SS}}$ is a convex set. Therefore, the optimization problem is a convex optimization problem. The three constraints are explained as follows.

(1) Capacity constraints: when transmitting from i to j, the rate of the network information flow transferred to the sink node t is not more than the sum of the rate of network coded packets injected to all the hyperarcs including the link (i, j).

(2) Flow constraints C_F: in a multicast session, the input-output relationship of the information flow rate at each node must meet the information flow balance constraints. Here the required fixed multicast rate is denoted by R.

(3) Scheduling constraints: in order to decompose the NC subgraph into a convex set of effective scheduling, the NC subgraph Z must be located in the stable set polytope $\mathrm{CH_{SS}}$ of a conflict graph.

Theoretical studies have shown that the optimal NC subgraph can be obtained by solving the optimization problems above. Then, the multicast transmission can meet all the constraints through RNC [2, 3]. Note that in this optimization framework we only consider the case using NC in a single multicast stream. The case of multiple multicast streams is thus similar to a single multicast stream. The case of using NC among multicast streams is left to future studies.

4. Optimization Algorithm

The global optimization in the stable set polytope $\mathrm{CH_{SS}}$ has been proved to be a NP-hard problem [12]. Therefore, there is no efficient solution to solve it even with a centralized algorithm. We thus propose to construct a heuristic subgradient algorithm by using Lagrangian dual theory.

4.1. Primal-Dual Subgradient Algorithm. Now let $\lambda = (\lambda_{ij}^{(t)})$ denote Lagrangian multipliers. Using Lagrangian relaxation, we can construct a dual function of the objective function of the original problem by moving the inequality constraint (7) into the objective function; that is,

$$q(\lambda) = \min_{X \in C_F, Z \in \mathrm{CH_{SS}}} \mathscr{L}(X, Z, \lambda), \tag{10}$$

where $X = (x_{ij}^{(t)})$ and C_F is defined as a flow constraint in (8). Here $\mathscr{L}(X, Z, \lambda)$ represents the corresponding Lagrangian function with λ, which is derived as

$$\begin{aligned}
&\mathscr{L}(X, Z, \lambda) \\
&= \sum_{(i,J) \in \mathscr{A}} f_{iJ}(z_{iJ}) \\
&\quad + \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{j \in N(i)} \lambda_{ij}^{(t)} \left(x_{ij}^{(t)} - \sum_{\{J | j \in J, J \subset N(i)\}} z_{iJ} \right) \\
&= \sum_{i \in \mathcal{N}} \sum_{J \subset N(i)} f_{iJ}(z_{iJ}) + \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{j \in N(i)} \lambda_{ij}^{(t)} x_{ij}^{(t)} \\
&\quad - \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{j \in N(i)} \lambda_{ij}^{(t)} \sum_{\{J | j \in J, J \subset N(i)\}} z_{iJ} \\
&= \sum_{i \in \mathcal{N}} \sum_{J \subset N(i)} f_{iJ}(z_{iJ}) + \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{j \in N(i)} \lambda_{ij}^{(t)} x_{ij}^{(t)} \\
&\quad - \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{J \subset N(i)} z_{iJ} \sum_{j \in J} \lambda_{ij}^{(t)} \\
&= \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{j \in N(i)} \lambda_{ij}^{(t)} x_{ij}^{(t)} + \sum_{i \in \mathcal{N}} \sum_{J \subset N(i)} \zeta_{iJ} z_{iJ} \\
&\quad - \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{J \subset N(i)} z_{iJ} \sum_{j \in J} \lambda_{ij}^{(t)} \\
&= \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{j \in N(i)} \lambda_{ij}^{(t)} x_{ij}^{(t)} \\
&\quad + \sum_{i \in \mathcal{N}} \sum_{J \subset N(i)} z_{iJ} \left(\zeta_{iJ} - \sum_{t \in T} \sum_{j \in J} \lambda_{ij}^{(t)} \right).
\end{aligned} \tag{11}$$

In (11), note that the last line can be decomposed into two parts that are coupled with Lagrangian multiplier $\lambda_{ij}^{(t)}$.

Step 1. initializing λ and set $n = 1$;
Step 2. solving the sub-problem 1 in (12) to obtain $\widehat{X}[n]$;
Step 3. solving the sub-problem 2 in (12) to obtain $\widehat{Z}[n]$;
Step 4. calculating the optimal value $f^*[n]$ of the main problem with (17);
Step 5. if a stopping criterion is satisfied or the maximum number of iterations is reached,
then the calculation is terminated; otherwise, calculating $\lambda[n+1]$ by (14), set $n = n + 1$ and then return to Step 2.

Algorithm 1: The primal-dual subgradient iterative optimization algorithm.

Substituting (11) into (10), we then can decompose this problem into two subproblems; that is,

$$\begin{aligned} q(\lambda) &= \min_{X,Z} \mathscr{L}(X, Z, \lambda) \\ &= \underbrace{\min_{X \in C_F} \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{j \in N(i)} \lambda_{ij}^{(t)} x_{ij}^{(t)}}_{\text{sub-problem 1}} \\ &\quad + \underbrace{\min_{Z \in \mathrm{CH}_{SS}} \sum_{i \in \mathcal{N}} \sum_{J \subset N(i)} z_{iJ} \left(\zeta_{iJ} - \sum_{t \in T} \sum_{j \in J} \lambda_{ij}^{(t)} \right)}_{\text{sub-problem 2}}. \end{aligned} \tag{12}$$

Then, the dual problem of the original problem becomes

$$\begin{aligned} \text{maximize} \quad & q(\lambda) = \min_{X,Z} \mathscr{L}(X, Z, \lambda) \\ \text{s.t.} \quad & \lambda \geq 0. \end{aligned} \tag{13}$$

To solve this dual problem, we use the projected subgradient algorithm with the iterative rule

$$\lambda[n+1] = \max\left(\left(\lambda[n] + \delta[n]\, \mathscr{g}[n] \right), 0 \right), \tag{14}$$

where $\delta[n]$ is a suitable step size and $\mathscr{g}[n] = (g_{ij}^{(t)}[n])$ is a subgradient at step n, which is computed by

$$g_{ij}^{(t)}[n] = \widehat{x}_{ij}^{(t)}[n] - \sum_{\{J \mid j \in J, J \subset N(i)\}} \widehat{z}_{iJ}[n]. \tag{15}$$

In (15), $\widehat{x}_{ij}^{(t)}[n]$ and $\widehat{z}_{iJ}[n]$ are the solutions of subproblems 1 and 2, respectively, at step n.

Note that the computation of each subgradient $g_{ij}^{(t)}[n]$ is only related to those local variables, that is, $\widehat{x}_{ij}^{(t)}[n]$ and $\widehat{z}_{iJ}[n]$. Therefore, all the computations can be carried out within a single-hop for each node. It indicates that a node based algorithm can be carried out in a distributed way without collecting the entire information of the network. That is, for updating Lagrangian multiplier $\lambda_{ij}^{(t)}$ by (14) and (15), each node only needs to communicate with its neighbor nodes rather than other nodes within two hops or more.

However, the intermediate solutions $\widehat{x}_{ij}^{(t)}[n]$ and $\widehat{z}_{iJ}[n]$ might not be optimal. We thus use a primary recovery technique to enable near-optimal results [13]. For simplifying the description, the optimal results at step n for each subproblem are denoted by

$$\begin{aligned} \widehat{X}[n] &= \left(\widehat{x}_{ij}^{(t)}[n] \right) \\ \widehat{Z}[n] &= \left(\widehat{z}_{iJ}[n] \right). \end{aligned} \tag{16}$$

The core idea of the primary recovery technique is to compute the convex set of all historical results as the current optimal solution of the main problem. In detail, the optimal solution of the main problem in step n can be obtained through the following rules:

$$\begin{aligned} X^*[n] &= \left(\left(\widehat{x}_{ij}^{(t)}[n] \right)^* \right) = \sum_{l=1}^{n} \tau_l[n]\, \widehat{X}[l] \\ Z^*[n] &= \left(z_{iJ}^*[n] \right) = \sum_{l=1}^{n} \tau_l[n]\, \widehat{Z}[l] \\ f^*[n] &= f\left(Z^*[n] \right) = \sum_{(i,J) \in \mathscr{A}} f_{iJ}\left(z_{iJ}^*[n] \right), \end{aligned} \tag{17}$$

where

$$\forall \tau_l[n] \geq 0, \quad \sum_{l=1}^{n} \tau_l[n] = 1. \tag{18}$$

Finally, the primal-dual subgradient iterative optimization algorithm is summarized in Algorithm 1.

Larsson et al. have proved that the iterative algorithm can guarantee to converge to the global optimal solutions of the main problem if setting $\delta[n] = a/(b+n)$ (where $a > 0$ and $b \geq 0$) and $\tau_l[n] = 1/n$ [13]. Other options also can be found in [3]. Considering the stopping criterion of the algorithm, it can be designed according to the gap between the optimal results of the main problem generated at each iteration and those at the last iteration. For example, the stopping criterion can be designed as $|f^*[n] - f^*[n-1]| \leq \varepsilon$, where ε is the accuracy requirement.

4.2. Solution of the Subproblems. As shown in Algorithm 1, the key of the algorithm lies in solving the two subproblems. First, let us focus on subproblem 1 in (12); that is,

$$\min_{X \in C_F} \sum_{t \in T} \sum_{i \in \mathcal{N}} \sum_{j \in N(i)} \lambda_{ij}^{(t)} x_{ij}^{(t)}. \tag{19}$$

Note that $\lambda_{ij}^{(t)}$ is nonnegative. We thus can consider $\lambda_{ij}^{(t)}$ as an equivalent length of the corresponding communication link

While the vertex set $\mathcal{T}$ contains at least one node do
Step 1. collect one vertex v from the collection $\mathcal{T}$ randomly
Step 2. add the vertex v to the stable set S
Step 3. remove vertex v and all the vertices connected to it from the collection $\mathcal{T}$
end while
return S as the random maximal stable set

Algorithm 2: The sampling algorithm of random maximal stable set.

(i, j). Therefore, this subproblem can be treated as a classic shortest path searching problem: searching for the shortest path from the source node to each sink node t, in which the length of the path is measured by $\lambda_{ij}^{(t)}$. Obviously, this problem can be solved by the traditional asynchronous distributed Bellman-Ford algorithm [14]. Therefore the solution of subproblem 1 can be distributed.

Now let us focus on subproblem 2 in (12); that is,

$$\min_{Z\in \mathrm{CH}_{\mathrm{SS}}} \sum_{i\in\mathcal{N}} \sum_{J\subset N(i)} z_{iJ} \left(\zeta_{iJ} - \sum_{t\in T} \sum_{j\in J} \lambda_{ij}^{(t)} \right). \tag{20}$$

Then, we transform it into a maximum problem:

$$\max_{Z\in \mathrm{CH}_{\mathrm{SS}}} \sum_{i\in\mathcal{N}} \sum_{J\subset N(i)} z_{iJ} \left(\sum_{t\in T} \sum_{j\in J} \lambda_{ij}^{(t)} - \zeta_{iJ} \right). \tag{21}$$

Additionally, we define the weight of each hyperarc corresponding to each node i as

$$w_{iJ} = \sum_{t\in T} \sum_{j\in J} \lambda_{ij}^{(t)} - \zeta_{iJ}, \quad \forall J \subset N(i). \tag{22}$$

Note that both $\lambda_{ij}^{(t)}$ and ζ_{iJ} are local variables which can be collected and computed within one-hop of node i. Therefore, w_{iJ} can be obtained via a node based distributed algorithm. Because Z belongs to a stable set polytope $\mathrm{CH}_{\mathrm{SS}}$ and w_{iJ} can also be viewed as an equivalent weight of z_{iJ}, this subproblem is a typical maximum weighted stable set problem in the scheduling optimization [15]. The difficulty of this subproblem lies in searching for the entire $\mathrm{CH}_{\mathrm{SS}}$, which results in a NP-hard problem [12]. This means that, for both centralized algorithms and distributed algorithms, the bottleneck is the solution of the scheduling subproblem.

In order to avoid the NP-hard complexity caused by searching for the entire $\mathrm{CH}_{\mathrm{SS}}$, a greedy algorithm named GWMIN was introduced in [10], in which the core idea of updating weights heuristically comes from [16]. The key steps of GWMIN are to update the weights in a heuristic way and search for a maximum stable set by certain rules, instead of searching for the entire $\mathrm{CH}_{\mathrm{SS}}$, at each iteration. However, the GWMIN has poor accuracy and robust performances. To overcome these drawbacks, this paper proposes a random sampling $\mathrm{CH}_{\mathrm{SS}}$ based algorithm, given a conflict graph, sampling K maximum stable sets $\mathrm{CH}_{\mathrm{SS}}^{K} \subset \mathrm{CH}_{\mathrm{SS}}$ randomly at each iteration, and solving subproblem 2 with $\mathrm{CH}_{\mathrm{SS}}^{K}$ random maximum stable sets. Now we use $I_t^{S(k)}$ to represent the incidence vector of the maximum stable set $S(k)$ obtained at the kth random sampling. Then we can achieve the approximate solution of (21) by solving the following optimization problem:

$$\begin{aligned} \max \quad & \sum_{i\in\mathcal{N}} \sum_{J\subset N(i)} z_{iJ} w_{iJ} \\ \text{s.t.} \quad & Z = (z_{iJ}) = \sum_{k=1}^{K} \rho_i I_t^{S(k)} \\ & \forall \rho_i \geq 0, \quad \sum_{i=1}^{K} \rho_i = 1. \end{aligned} \tag{23}$$

Apparently, the optimization problem (23) is a linear programming problem so that an efficient polynomial algorithm such as the simplex method can be adopted. Therefore, using the K maximum stable sets obtained by sampling $\mathrm{CH}_{\mathrm{SS}}$ randomly, we can transfer the original NP-hard problem into a general linear programming problem to obtain an approximate solution. Finally, given a conflict graph $\mathcal{G} = (\mathcal{T}, \mathcal{B})$, the random sampling algorithm for each maximum stable set is presented in Algorithm 2.

As analyzed before, the computation of w_{iJ} can be distributed. Furthermore, the random sampling algorithm for maximum stable sets also can be implemented in a distributed algorithm proposed in [17]. After collecting the necessary information of w_{iJ} and K maximal stable sets in a distributed way, problem (23) can be solved by a centralized algorithm. As a result, on the proposed solution for problem (12), only this part is centralized, and all of the others can be distributed.

5. Simulation Results

In order to analyze the performances of the proposed algorithm, we developed a simulation package in MATLAB. To solve problem (23), we used CVX, a package for specifying and solving convex programs [18]. For each simulation, we used a random network topology model in which all nodes were distributed within a rectangular area randomly and uniformly with a unit density. Any two nodes are viewed as reachable when the distance between them is less than a certain communication radius r. We used multicast scenarios with one session and two sink nodes in simulations. We also used an auxiliary interference model to model a conflict graph. We selected the leftmost node as the source node and

Table 1: The average optimal results over 100 random network topologies.

Algorithms	Number of nodes			
	5	10	15	20
The optimal algorithm	1.41	2.96	4.44	5.64
The GWMIN algorithm	1.91	4.63	5.45	8.34
The K-RMSS algorithm	1.42 ($K = 5$)	3.02 ($K = 20$)	4.47 ($K = 30$)	6.89 ($K = 60$)
The gain of performance	26%	35%	22%	17%

the rightmost two nodes as sink nodes for each multicast scenario within a random generated network topology.

5.1. Performance Comparisons. We simulate the following three algorithms: a full centralized algorithm indicating the global optimal results; the GWMIN algorithm presented in [10]; and the K random maximum stable sets (K-RMSS) based algorithm proposed in this paper.

For a fair comparison, we used the same stopping criterion and iterative parameters for both GWMIN and K-RMSS. The step size is set as $\delta[n] = 0.6/(1+n)$ and the stopping criterion is set as $|f^*[n] - f^*[n-1]| \leq 0.01$. We simulated four different random scenarios with the numbers of nodes of 5, 10, 5, and 20, respectively. Note that the number of nodes in a conflict graph is the number of hyperarcs in a hypergraph, which is growing exponentially with the number of neighbors for each node. Accordingly, in order to reduce the number of neighbors for each node, r is set to 1.8, in scenarios where the number of nodes is not more than 15; r is set to 1.6 in scenarios where the number of nodes is 20. In addition, we also adopted different K for simulations in scenarios with different numbers of nodes. Actually, the parameter K has a great effect on the optimization performance. It will be discussed in the next section. For each case, we solved the optimization problem with different algorithms for 100 random topologies and then computed the average values of optimization results, as shown in Table 1.

The last row in Table 1 represents the normalized gain of the K-RMSS algorithm over the GWMIN algorithm. It indicates that K-RMSS outperforms GWMIN significantly. In most cases, this gain can be more than 20%. Further, we can see that the performance of K-RMSS is almost close to the optimal result when the number of nodes is 5, 10, and 15. The main reason is that GWMIN selects only one stable set with relative large weights at each iteration while K-RMSS selects K stable subsets randomly at each iteration. Therefore, after a certain number of iterations, K-RMSS can better approximate the global optimal results. Comparing with GWMIN, the complexity of K-RMSS is mainly to select more maximal stable sets for the optimization. Fortunately, this is a linear programming problem, which can be solved efficiently.

Finally, note that the performance of K-RMSS is far from the global optimization result when the number of nodes is 20. This is because the number of hyperarcs in a conflict graph is growing exponentially with the network size. To achieve a good performance, therefore, the parameter K needs to increase with the increase of the network size. The impacts of K on the performance will be analyzed in the next subsection.

Table 2: Comparisons of average running time of one iteration.

GWMIN	K-RMSS			
	$K = 20$	$K = 40$	$K = 60$	$K = 80$
0.27 s	0.46 s	0.48 s	0.50 s	0.55 s

5.2. Effects of the Parameters. For studying the impacts of the parameter K and the step size, we have chosen a multicast scenario with the number of nodes of 20 and random topologies. First, let us focus on the impacts of the parameter K. For this case, the step size is still set to $\delta[n] = 0.6/(1+n)$, the communication radius r is set to 1.6, and the parameter K in K-RMSS is set to 40, 50, and 80, respectively. The simulation results for 1000 iterations are shown in Figure 3.

From Figure 3, we can see that the performance of K-RMSS is always better than GWMIN. This is because the size of the maximum stable sets employed by K-RMSS is much larger than that employed by GWMIN, which results in the size of the search space in K-RMSS being also much larger than that in GWMIN. Figure 3 also shows that the performance of K-RMSS can be improved by increasing the value of the parameter K. This is also because the size of the search space in K-RMSS increases with the increase of the parameter K. Therefore, the larger the parameter K is, the better the performance it can reach.

Additionally, Figure 3 shows that both K-RMSS and GWMIN have converged within 100 iterations. To compare their efficiencies with each other, we have recorded the running times of 1000 iterations for each simulation and then computed the average running time of one iteration for both K-RMSS and GWMIN, as concluded in Table 2. The simulations have been carried out under the following software and hardware environments: CVX (version 1.22), MATLAB (version 8.3.0), Windows 7 OS, 3.4 G CPU, and 8 G RAM.

From Table 2, we can see that the average running time of K-RMSS with $K = 80$ is twice that of GWMIN due to larger search space. However, from Figure 3, we also find that by some improved methods the 80-RMSS can converge within about 50 iterations, which is nearly half of that in GWMIN. It indicates that both GWMIN and 80-RMSS can have similar efficiency for the convergence. Moreover, Table 2 shows that the average running time decreases with the decrease of the value of K. Therefore, the parameter K can provide a good way for the tradeoff between the performance and the efficiency.

Now let us focus on the influence of the step size on the performance of the algorithms. For this case, we modified the

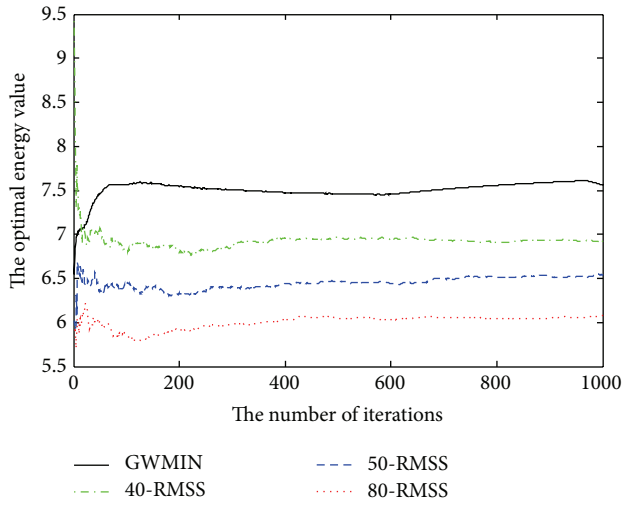

Figure 3: The influence of K on the performance of K-RMSS.

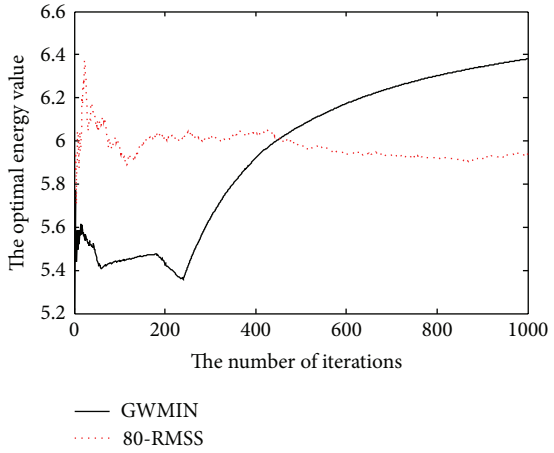

Figure 4: The influence of the step size on the performance.

step size to $\delta[n] = 0.1/(1+n)$ and kept the other parameters similar to those in the above simulations. Then, we carried out simulations for GWMIN and K-RMSS with $K = 80$. The simulation results are shown in Figure 4.

From Figure 4, we can see that K-RMSS still can converge within 100 iterations, which is similar to that in Figure 3. However, Figure 4 shows that GWMIN cannot converge even within 1000 iterations. It indicates that GWMIN is much more sensitive to the step size than K-RMSS. Therefore, comparing with GWMIN, K-RMSS can adapt to the step size and achieve better robustness. It means that, to achieve fast and good convergence, GWMIN must choose the step size carefully in advance while K-RMSS can set it randomly within a reasonable range. Therefore, K-RMSS is suitable real-time services much more than GWMIN. Finally, as shown in Figure 4, the performances of GWMIN are better than K-RMSS before 400 iterations due to the randomness of initial values (i.e., the initial value of λ is set randomly in the simulation). From the trend of the convergent results, however, we can see that K-RMSS outperforms GWMIN significantly.

6. Conclusions

In wireless multihop networks, there is a challenge on minimizing the energy of transmissions with guaranteeing a fixed and qualified data rate for real-time multicast services. By addressing this issue, using a conflict graph model, we construct a joint network coding and scheduling optimization framework for modeling this optimization problem. Using Lagrangian decompositions, we also propose a primal-dual subgradient algorithm to solve it. This method can transform a NP-hard scheduling subproblem into a normal linear programming problem to obtain an approximate solution by sampling K maximal stable sets randomly. Moreover, most calculations of the proposed algorithm can be carried out by a distributed algorithm, and only the solution of a linear programming problem needs to be done by a centralized algorithm. Simulation results show that the performance of this algorithm outperforms the existing algorithms significantly. The studies also show that the parameter K provides a good way for reaching a tradeoff between the efficiency and the performance. Finally, compared with the existing algorithms, the proposed algorithm can converge by setting the step size of iterations randomly, which results in better robustness.

Since the parameter K, the initial values, and the step size have significant impacts on performance of the proposed algorithm, how to set the most suitable parameters and update them online needs more research. Finally, we would like to point out that recent studies have shown that intersession NC can achieve better throughput performance than intrasession NC [19, 20]. Therefore, how to integrate intersession NC into the optimization framework is another worthy research direction in the future.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for valuable comments and suggestions for improving the quality of this paper. This work was supported in part by National Natural Science Foundation of China (61001068).

References

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

[2] T. Ho, M. Medard, R. Koetter et al., "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[3] D. S. Lun, N. Ratnakar, M. Medard et al., "Minimum-cost multicast over coded packet networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2608–2623, 2006.

[4] Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1906–1918, 2005.

[5] S. H. Lee and S. Vishwanath, "Distributed rate allocation for network-coded multicast networks," *IEEE Communications Letters*, vol. 17, no. 11, pp. 2204–2207, 2013.

[6] Y. Wu and S.-Y. Kung, "Distributed utility maximization for network coding based multicasting: a shortest path approach," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1475–1488, 2006.

[7] J. J. Jaramillo, R. Srikant, and L. Ying, "Scheduling for optimal rate allocation in ad hoc networks with heterogeneous delay constraints," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 979–987, 2011.

[8] J. J. Jaramillo and R. Srikant, "Optimal scheduling for fair resource allocation in Ad hoc networks with elastic and inelastic traffic," *IEEE/ACM Transactions on Networking*, vol. 19, no. 4, pp. 1125–1136, 2011.

[9] K. Rajawat and G. B. Giannakis, "Joint scheduling and network coding for multicast in delay-constrained wireless networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6186–6196, 2011.

[10] D. Traskov, M. Heindlmaier, M. Medard, and R. Koetter, "Scheduling for network-coded multicast," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1479–1488, 2012.

[11] G. P. Fettweis, "The tactile internet: applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, NY, USA, 1979.

[13] T. Larsson, M. Patriksson, and A.-B. Stroemberg, "Ergodic, primal convergence in dual subgradient schemes for convex programming," *Mathematical Programming*, vol. 86, no. 2, pp. 283–312, 1999.

[14] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 1992.

[15] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[16] S. Sakai, M. Togasaki, and K. Yamazaki, "A note on greedy algorithms for the maximum weighted independent set problem," *Discrete Applied Mathematics*, vol. 126, no. 2-3, pp. 313–322, 2003.

[17] S. Basagni, "Finding a maximal weighted independent set in wireless networks," *Telecommunication Systems*, vol. 18, no. 1–3, pp. 155–168, 2001.

[18] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," September 2013, http://cvxr.com/cvx.

[19] M. Heindlmaier, D. S. Lun, D. Traskov, and M. Medard, "Wireless inter-session network coding—an approach using virtual multicasts," in *Proceedings of the IEEE International Conference on Communications (ICC '11)*, pp. 1–5, Kyoto, Japan, June 2011.

[20] M. Esmaeilzadeh and N. Aboutorab, "Inter-session network coding for transmitting multiple layered streams over single-hop wireless networks," in *Proceedings of the IEEE Information Theory Workshop (ITW '14)*, pp. 401–405, Hobart, Australia, November 2014.