Scheduling for Network Coded Multicast: A Distributed Approach

Michael Heindlmaier, Danail Traskov, Ralf Kötter
Institute for Communications Engineering
Technische Universität München
Munich, Germany
Email: {michael.heindlmaier, danail.traskov}@tum.de

Muriel Médard
Research Laboratory of Electronics
MIT
Cambridge, MA
Email: medard@mit.edu

***Abstract*—We address the problem of maximizing the throughput for network coded multicast traffic in a wireless network in the bandwidth limited regime. For the joint scheduling and subgraph selection problem, we model valid network configurations as stable sets in an appropriately defined conflict graph. The problem formulation separates the combinatorial difficulty of scheduling from the arising optimization problem and facilitates the application of less complex scheduling policies. Lagrangian relaxation gives rise to two subproblems, a multiple shortest path, and a maximum weight stable set (MWSS) problem. For the latter we propose a greedy approach which can be computed in a distributed fashion, thus yielding a fully decentralized algorithm. Simulation results show that our technique is nearly optimal and outperforms heuristics such as orthogonal scheduling by a large margin.**

I. Introduction

In wireless multi-hop networks a promising strategy for multicasting to a group of receivers is to use random linear network coding [1]. While randomly mixing packets at intermediate nodes and decoding at the multicast destinations can be done under virtually any underlying multi-access and routing scheme, there is substantial evidence that one can capitalize on significant gains by optimizing these layers for network coded traffic.

As demonstrated in [2] the network coding subgraph, which relates the node transmission rates to the effective link flows, can be cast into a linear or convex program. This is a fairly tractable problem and furthermore by applying a subgradient or primal-dual algorithm, respectively, the computation can be distributed. Simulations demonstrated significant gains for network coding over an optimized subgraph as compared to applying link-based FEC [2].

Another step towards optimizing network performance is the approach taken in [3] to include the medium-access constraints in the joint optimization. There, the authors constructed a conflict graph model based on conflicting hyperarcs to identify valid network configurations. The notion of a conflict graph (see also [4] for a related approach) is used to describe interference-free transmissions which correspond to stable sets[1]. The problem formulation using conflict graphs is well known e.g. for routed traffic in [5], for scheduling in switches in [6], for network code construction in [7], and in the context of Banyan networks in [8]. A new scheduling formulation is required, as the nature of network coded flows is not captured well by classical link based scheduling. Simulation results demonstrate significant gains when the MAC-layer is included in the optimization as compared to [2], [9], where interference is not treated by assuming that each node in the network transmits on its own orthogonal channel.

The main reason to consider scheduling algorithms, despite the increase in complexity, is the very poor performance of scheduling heuristics such as orthogonal scheduling, particularly where bandwidth is scarce. In fact, it was shown [3] that by using ad-hoc scheduling heuristics of that kind, we lose so much throughput that it becomes the most significant limiting factor of the system.

The approach in [3], while jointly optimizing the channel access and the network coding subgraph is based on two assumptions. Firstly, the computation has to be carried out by a central processor who has knowledge of the entire network topology, and secondly the optimization is solved over the entire stable set polytope of the conflict graph. Both conditions are rather stringent and likely not available in practical mobile ad-hoc networks.

The focus of our work and the contribution beyond [3] is to relax these assumptions by proposing *decentralized* and *low-complexity* algorithms. To distribute the computation across the network we propose a subgradient algorithm applied to a Lagrangian relaxation of the optimization problem. In order to reduce the computational complexity we experimentally show that simple techniques, such as greedily choosing maximal stable sets, perform well and for networks of moderate size are not far from the optimum.

The works by [10], [11] and the game theoretic formulation by [12] consider related problems, but using very different approaches. In [11], the authors consider a similar channel model and determine conflict-free network realizations with power assignment a priori. Also the works [10], [12] construct conflict-free transmissions by power-control. We, however, consider the case when all nodes have equal power, so the number of valid transmission configurations is finite. Our contributions, in particular those that go beyond [3] are, in summary, that we

- provide a formulation where the combinatorial difficulty

[1]Some authors prefer to use the term independent sets.

of the scheduling problem is isolated from the optimization aspects of the system,
- develop less complex scheduling techniques both for centralized and decentralized optimization,
- present a heuristic, yet *fully distributed* algorithm for jointly optimizing MAC and subgraph,
- and by means of simulations demonstrate good and sometimes even close to optimal performance of the suggested techniques.

The remainder of this paper is organized as follows: In Section II we give a short overview of the network model. We review the centralized optimization problem in Section III while Section IV deals with distributed solutions via dual decomposition. Section V shows simulation results for optimal and low-complexity scheduling techniques. We conclude the paper in Section VI.

II. Network Model

We model a wireless network by a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{A}$ the set of hyperarcs. A hyperarc, a generalization of an edge, is a pair (i, J), with transmitting node $i \in \mathcal{N}$ and nonempty receive set $J \subset \mathcal{N}$.

The network operates in slotted time and with constant-length packets. Node i injects packets into hyperarc J according to a process[2] with rate z_{iJ}; however, owing to erasures the packets may not be received by all nodes in J but rather by a subset $K \subseteq J$. Thus the arrival rate z_{iJK} is the average rate at which a packet is received *precisely* by subset K. Therefore we have $z_{iJ} = \sum_{K \subset J} z_{iJK}$. We will assume throughout the paper that $p_{iJK} = z_{iJK}/z_{iJ}$, the probability that a packet injected on (i, J) is received precisely by subset K, is constant. We refer to the injection rate vector $\mathbf{z} = (z_{iJ})_{(i,J)\in\mathcal{A}}$ as the *network coding subgraph*.

To achieve the capacity of the subgraph, i.e. its min-cut with respect to each multicast receiver, we shall use random linear network coding [1]. Coding is restricted to packets belonging to the same multicast source (intra-session).

When a node i transmits it can reach a set of nodes $N(i) \subset \mathcal{N}$, which we call its one-hop-neighborhood. To schedule transmissions to certain subsets of its neighbors, we create an outgoing hyperarc for every subset of $N(i)$, excluding the empty set, leading to $2^{|N(i)|} - 1$ hyperarcs with source i. We assume that nodes outside this neighborhood can neither receive the packet nor experience interference when node i is transmitting

Scheduling transmissions for network coded traffic differs from classical link-based scheduling in that we activate hyperarcs instead of links [3]. In this paper a conflict graph was constructed for scheduling non-conflicting simultaneous transmissions assuming hard interference constraints. The conflict graph $\mathcal{G}$ of a hypergraph $\mathcal{H}$ is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex in $\mathcal{V}$ stands for a hyperarc in the underlying hypergraph. Throughout this work, the term 'node' refers to elements of the hypergraph $\mathcal{H}$ while 'vertex' is used for an element of the conflict graph $\mathcal{G}$ like in [4]. Two vertices in $\mathcal{G}$ are connected if the underlying hyperarcs in $\mathcal{H}$ conflict. We shall define precisely the notion of conflict below.

We assume half-duplex constraints and secondary interference, i.e. if a node is in range of more than one simultaneous transmissions then all packets are lost. Under these assumptions, two hyperarcs (i_1, J_1) and (i_2, J_2) do *not* conflict if:

1) $i_1 \neq i_2$
2) $i_1 \notin J_2, i_2 \notin J_1$
3) $J_1 \cap N(i_2) = \emptyset$, and $J_2 \cap N(i_1) = \emptyset$.

Note that even in the absence of interference packets can be lost owing to erasures. This is taken into account by the loss probabilities p_{iJK}.

The rate region of the network consists of all points that can be achieved by time sharing over valid network configurations. Such a valid, interference-free configuration corresponds to a stable set in the conflict graph. Thus the rate region corresponds to the convex hull of the incidence vectors of all maximal stable sets of $\mathcal{G}$, i.e. it is equivalent to the stable set polytope $P_{STAB}(\mathcal{G})$ of the conflict graph [3]. For a detailed description of the conflict graph model we refer to [3].

III. The Optimization Problem

Using conflicting hyperarcs as a model for scheduling constraints, we can formulate a linear program that jointly optimizes the network coding subgraph and the underlying schedule. For maximal packet throughput per timeslot, denoted R, for the multicast from a source s to a set of receivers T the problem was derived in [3] and takes on the form

$$\max_{R,\mathbf{x},\mathbf{z}} \quad R \qquad \text{subject to} \tag{1}$$

Capacity constraints:

$$\sum_{j\in K} x_{ij}^{(t)} \leq \sum_{J\subset N(i)} z_{iJ} b_{iJK}, \tag{2}$$
$$\forall\ i \in \mathcal{N}, K \subset N(i), t \in T,$$

Flow constraints $\mathbf{F}$:

$$\sum_{j\in N(i)} x_{ij}^{(t)} - \sum_{j|i\in N(j)} x_{ji}^{(t)} = \begin{cases} R & i = s, \\ -R & i = t, \\ 0 & \text{else}, \end{cases} \tag{3}$$
$$\forall\ i \in \mathcal{N}, t \in T,$$

$$(R, \mathbf{x}) \geq 0, \qquad \mathbf{x} = (x_{ij}^{(t)})_{i\in\mathcal{N}, j\in N(i), t\in T} \tag{4}$$

Scheduling constraints:

$$\mathbf{z} \in P_{STAB}(\mathcal{G}),\ \mathbf{z} = (z_{iJ})_{(i,J)\in\mathcal{A}} \tag{5}$$

where we define

$$b_{iJK} = \sum_{\{L\subset J | L\cap K\neq\emptyset\}} p_{iJL}. \tag{6}$$

Here, the capacity constraints (2) relate the network coding subgraph $\mathbf{z}$ to the packet flow $\mathbf{x}$. The conservation laws (3) and (4) define flow polytopes for each multicast destination. Capacity and flow conditions are essentially the same as in [2].

[2]The only assumption about the process is that it is ergodic and thus has a time average.

The scheduling constraints (5) account for the effects of multi-access interference and half-duplex constraints. Optimizing over the stable set polytope P_{STAB} is an NP-hard problem [13], while the other linear constraints are relatively simple. Therefore, we isolate the combinatorial difficulty of scheduling from the optimization of injection rates and flows.

IV. Decentralized Formulation

While the above formulation (1) - (5) allows us to optimize jointly subgraph and schedule, the computation, as it stands, has to be carried out in a centralized fashion. In this section, we solve this optimization problem with a subgradient-based algorithm. To that end, we construct its Lagrangian relaxation by dualizing the capacity constraints in (2).

$$L(R,\mathbf{z},\mathbf{x},\boldsymbol{\lambda}) = R + \sum_{t\in T}\sum_{i\in\mathcal{N}}\sum_{K\subset N(i)} \lambda_{iK}^{(t)} \left(\sum_{J\subset N(i)} z_{iJ} b_{iJK} - \sum_{j\in K} x_{ij}^{(t)} \right), \tag{7}$$

$$\lambda_{iK}^{t} \geq 0, \qquad \mathbf{z} = (z_{iJ}), \quad \mathbf{x} = (x_{ij}^{(t)}), \quad \boldsymbol{\lambda} = (\lambda_{iK}^{(t)})$$

The dual function takes on the form

$$\begin{aligned} q(\boldsymbol{\lambda}) \quad &= \max_{R,\mathbf{z},\mathbf{x}} L(R,\mathbf{z},\mathbf{x},\boldsymbol{\lambda}) \\ \text{subject to} \quad &\mathbf{z}\in P_{STAB}, \quad (R,\mathbf{x})\in\mathbf{F}, \end{aligned} \tag{8}$$

where the flow polytope $\mathbf{F}$ (8) is defined by the flow conservation constraints in (3) and (4).

The dual function decomposes into two subproblems, coupled by the Lagrangian multipliers $\lambda_{iK}^{(t)}$, as follows

$$q(\boldsymbol{\lambda}) = \underbrace{\max_{(R,\mathbf{x})\in\mathbf{F}} \left(R - \sum_{t\in T}\sum_{i\in\mathcal{N}}\sum_{K\subset N(i)} \lambda_{iK}^{(t)} \sum_{j\in K} x_{ij}^{(t)} \right)}_{\text{Subproblem 1}} + \underbrace{\max_{\mathbf{z}\in P_{STAB}} \left(\sum_{t\in T}\sum_{i\in\mathcal{N}}\sum_{K\subset N(i)} \lambda_{iK}^{(t)} \sum_{J\subset N(i)} z_{iJ} b_{iJK} \right)}_{\text{Subproblem 2}}. \tag{9}$$

A. Subproblem 1: Multiple Shortest Paths

The first subproblem in (9) is a multiple shortest path problem due to the constraints in $\mathbf{F}$. The flow polytope $\mathbf{F}$ itself yields no upper bound of the maximal throughput, however, we know that $(R,\mathbf{x}) \leq 1$ since $z_i = \sum_{J\subset N(i)} z_{iJ} \leq 1$ due to (5) and $x_{ij}^{(t)} \leq z_i$ due to (2). So we impose the additional constraint $0 \leq (R,\mathbf{x}) \leq 1$, which does not restrict the feasible rate region. Rearranging the first part of (9) yields

$$(\hat{R},\hat{\boldsymbol{x}}) = \arg \max_{\substack{(R,\mathbf{x})\in\mathbf{F} \\ 0\leq(R,\mathbf{x})\leq 1}} \left(R - \sum_{t\in T}\sum_{i\in\mathcal{N}}\sum_{j\in N(i)} x_{ij}^{(t)} p_{ij}^{(t)} \right), \tag{10}$$

where we define $p_{ij}^{(t)} = \sum_{K\subset N(i), j\in K} \lambda_{iK}^{(t)}$.

We explain briefly the equivalence of (10) and shortest path problems: To maximize (10), we have to increase R and keep the right expression small. However, R and $\mathbf{x}$ are coupled by $\mathbf{F}$. For a fixed R, the task is to minimize the cost

$$\sum_{t\in T} \min_{\mathbf{x}} \sum_{i\in\mathcal{N}}\sum_{j\in N(i)} x_{ij}^{(t)} p_{ij}^{(t)}, \tag{11}$$

from s to each t providing end-to-end throughput R. (11) is a standard minimum-cost network problem [14, Section 5] without capacity constraints. Therefore, it suffices for each of the $|T|$ parallel optimization problems to find a single minimum-cost path from s to t. Each of the flows in the minimum-cost path carries the total throughput R as there is only one active path from s to t, so we can reformulate (10):

$$\max_{\substack{R \\ 0\leq R\leq 1}} \left(R(1 - \sum_{t\in T}\sum_{(i,j)\in\mathcal{P}_{st}} p_{ij}^{(t)}) \right) \tag{12}$$

with $\mathcal{P}_{st}$ denoting the collection of links (i,j) in the minimum-cost path from s to t. If the cumulative cost of all paths from source to all terminals is smaller than 1, the intermediate optimizers equal 1 for $\hat{R}$ and for those link flows $\hat{x}_{ij}^{(t)}$ with $(i,j)\in\mathcal{P}_{st}$. Otherwise, they are equal to zero.

To solve (12), $|T|$ shortest paths have to be determined, which can be solved distributedly, such as by asynchronous Bellman-Ford [15, Section 5]. As all $\lambda_{iK}^{(t)}$ are greater than zero by definition (7), no cycles with negative costs exist and shortest path algorithms converge [15, Section 5].

B. Subproblem 2: Maximum Weight Stable Set

Subproblem 2 can be rearranged to yield

$$\hat{\mathbf{z}} = \arg \max_{\mathbf{z}\in P_{STAB}} \sum_{i\in\mathcal{N}}\sum_{J\subset N(i)} z_{iJ} w_{iJ} \tag{13}$$

with weights $w_{iJ} = \sum_{K\subset N(i)} \left(b_{iJK} \sum_{t\in T} \lambda_{iK}^{(t)} \right)$. This is a maximum weight stable set (MWSS) problem, a well-known problem in the scheduling literature [5]. The computational complexity of the overall optimization problem is determined by this subproblem, which is NP-hard, in general [13].

C. Minimizing the Dual Function

The primal problem (1) is a linear program, so it has no duality gap and can be solved via the dual, which takes on the form

$$\min_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda}) \qquad \text{s.t.: } \boldsymbol{\lambda} \geq 0. \tag{14}$$

To determine the dual optimal value we can apply projected subgradient optimization [16, Section 8.9]. The update rule is

$$\boldsymbol{\lambda}[n+1] = \left(\boldsymbol{\lambda}[n] - \theta[n]\boldsymbol{\xi}[n] \right)^{+}, \tag{15}$$

where $(\centerdot)^{+} = \max(\centerdot, 0)$. $\theta[n]$ denotes a suitable stepsize and $\boldsymbol{\xi}[n]$ is a valid subgradient for $q(\boldsymbol{\lambda})$ at step n, given by

$$\boldsymbol{\xi}[n] = \left[\sum_{J\subset N(i)} \hat{z}_{iJ}[n] b_{iJK} - \sum_{j\in K} \hat{x}_{ij}^{(t)}[n] \right]_{(i,K)\in\mathcal{A}, t\in T} \tag{16}$$

with the locally available optimizers $\hat{\mathbf{x}}[n]$ and $\hat{\mathbf{z}}[n]$ from Subproblems 1 and 2 at step n.

D. Primal Recovery

One problem of subgradient methods is that the sequence $\hat{\mathbf{x}}[n]$ and $\hat{\mathbf{z}}[n]$ need not be feasible. Hence, we have to recover the primal solution. We adopt a technique proposed in [17], averaging the intermediate optimal values to obtain primal feasible solutions, i.e.

$$R^* = \frac{1}{N}\sum_{n=1}^{N}\hat{R}[n] \qquad \mathbf{x}^* = \frac{1}{N}\sum_{n=1}^{N}\hat{\mathbf{x}}[n] \qquad \mathbf{z}^* = \frac{1}{N}\sum_{n=1}^{N}\hat{\mathbf{z}}[n]$$

This method was shown to converge for a stepsize $\theta[n] = \frac{a}{b+cn}$ with $a > 0$, $b \geq 0$ and $c > 0$ [17]. Note, that the recovery rule does not require any additional message exchange, due to fact that all intermediate optimizers are available at the node performing the averaging (see [9] for a similar approach).

E. Distributed Implementation

For a decentralized solution, Subproblems 1 and 2 have to be computed distributedly. Finding a shortest path can be done with asynchronous Bellman-Ford [15, Section 5]. This algorithm converges in $O(|\mathcal{N}||E|)$ steps, with $|E| = \sum_{i\in\mathcal{N}}|N(i)|$. All messages for the $|T|$ parallel shortest paths computations can be exchanged in shared packets, thus not creating much additional overhead over single shortest path algorithms.

For subproblem 2, we propose a distributed decentralized approach in Section V-B. Necessary information exchange can be combined with the shortest path updates. Throughout this work, we assume that these messages are sent over error- and interference-free control channels. Since the update step (15) and primal recovery only include locally known variables at each node, the minimization (14) can be computed in a distributed fashion.

V. Simulations

For our simulations we use random network topologies, where nodes are randomly placed on a square region with unit node density. Two nodes are connected with a pair of directed links if their distance is below some radius of connectivity r, which we take to be 1.8, and a maximum of 5 neighbors is permitted. We consider a one-to-two multicast, from the leftmost node to the two rightmost nodes. Note, that even ignoring interference and half-duplex constraints [2], the number of hyperarcs scales exponentially with the number of neighbors - a necessity of modelling wireless broadcasting.

Transmissions are subject to erasures which are due to distance attenuation and Rayleigh fading. When a node transmits its neighbor at distance d will receive the packet correctly if $\gamma d^{-\alpha} \geq \beta$ where γ is the realization of a unit mean exponential variable, α is the path loss exponent, and β is the chosen SNR threshold, otherwise the packet is lost completely. This corresponds to an erasure probability $\epsilon = \Pr[\gamma d^{-\alpha} \leq \beta] = 1 - \exp(-\beta d^{\alpha})$. We will use for all simulations $\alpha = 2$ and $\beta = \frac{1}{4}$. We assume that all erasures to different receivers are independent, though this is not required by our model.

A. Centralized scheduling techniques

In [3], with the focus on centralized scheduling, three basic techniques were suggested. The highest throughput can be achieved by optimizing over P_{STAB}, i.e. solve precisely problem (1) - (5). Alternatively, one can instead optimize over a smaller polytope $\hat{P} \subset P_{STAB}$ with a simpler description. One such relaxation is the 2-hop constraint model, where all nodes within a 2-hop neighborhood of a transmitting node are required to be silent. Even more simply, one may allow at most one active node in the entire network in each time slot, thus making all transmissions orthogonal. We refer to these techniques, the performance[3] of which is shown in Fig. 1, as *optimal*-, *2-hop*-, and *orthogonal* scheduling, respectively.

These curves illustrate how one can trade-off throughput versus scheduling complexity. In general, it is computationally hard to optimize over the entire stable set polytope; on the other hand adopting a simplistic scheduling heuristic may lead to a severe loss in throughput. The rest of this Section deals with techniques which increase the throughput compared to orthogonal scheduling, while their computational complexity is roughly the same.

1) Iterative approach: Consider an iterative procedure which starts with a small initial collection of stable sets $\hat{P}$ and alternates between optimizing over $\hat{P}$ and adding more stable sets to $\hat{P}$ using sensitivity information from the previous optimization step (see [10] for a related approach). In particular, we start with $\hat{P}$ being the polytope for orthogonal scheduling and, after carrying out the optimization, identify the hyperarcs that constitute the min-cut. We find a certain number of stable sets containing these hyperarcs and add them to $\hat{P}$. Often, after one iteration we cannot further improve.

2) Random approach: As a second approach, a certain number of maximal stable sets is randomly sampled using a greedy algorithm. For a single maximal stable set, first an initial vertex in $\mathcal{G}$ is randomly chosen, then one of the non-adjacent vertices is selected randomly and added to the stable set. This procedure is pursued until no more vertex can be added. The resulting $\hat{P}$ is defined by the same number of stable sets as in the second step of the iterative approach.

Fig. 1 shows that these two techniques show nearly the same performance for larger networks. Although they cannot provide the optimal solution in general, the optimality gap is quite small. Interestingly, adding carefully chosen min-cut-increasing stable sets does not perform better than a simple random selection. Apparently, this is due to the fact that after orthogonal scheduling almost every hyperarc defines a min-cut. Packing these hyperarcs into stable sets thus almost corresponds to a random selection of stable sets [18].

Fig. 2 further illustrates the performance of the random approach. There, we consider networks with 10 nodes, which under our random model turn out to have on average 1200 maximal stable sets. We successively sample a number of stable sets S and compute the throughput $R[S]$ as a function

[3]The throughput under optimal scheduling was only computable for networks up to 13 nodes.

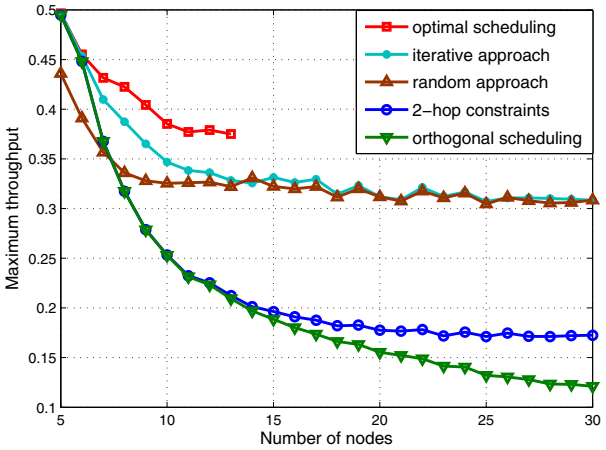

Fig. 1: Maximum throughput as a function of the number of nodes in the network, averaged over 300 random networks.

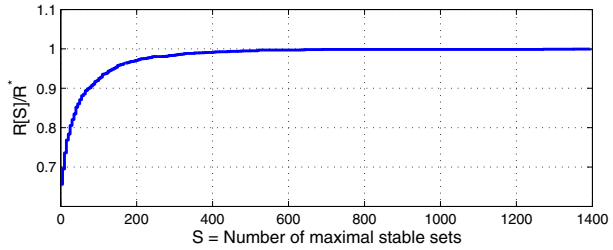

Fig. 2: Throughput as a function of the number of maximal stable sets, averaged over 100 random networks with 10 nodes.

of S. We report these throughputs normalized by the optimal throughput R^*, which is obtained by optimizing over the entire stable set polytope. We observe that 100 randomly determined maximal stable sets already yield a throughput which is on average 90% of the optimum, while 400 maximal stable sets usually yield the optimum. For a different size of the network we obtain very similar results. We conclude that, if the optimization can be computed by a central entity, it suffices to sample a rather small number of stable sets and to optimize over this polytope to obtain a close to optimal solution.

B. Distributed scheduling techniques

For a fully distributed operation, a decentralized maximum stable set algorithm is needed to solve (13) exactly. This problem is hard even for centralized computation. An algorithm with good results, though not optimal in general, is the greedy strategy in Algorithm 1 due to [19].

Algorithm 1 GMIN [19]

Require: A weighted conflict graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
1: **while** $\mathcal{V}$ contains at least one vertex **do**
2: Choose a vertex $v \in \mathcal{V}$
3: Add v to the stable set $\mathcal{S}$
4: Remove v and all its adjacent vertices from $\mathcal{V}$
5: **end while**
6: **return** A maximal stable set $\mathcal{S}$

Implementations of Algorithm 1 only differ with respect to which vertex is chosen at each step. We compare two rules:

1) Choose vertex $v \in \mathcal{V}$ with highest weight $w(v)$ (weights as defined by (13)) (decentralized greedy)
2) Choose vertex $v \in \mathcal{V}$ that maximizes $\frac{w(v)}{d_m(v)+1}$. The degree of the vertex v at step m is denoted by $d_m(v)$ and can be determined after removing the vertices contained in or adjacent to the stable set at step m (GWMIN [19]).

Rule 1) requires very little knowledge about the graph. A distributed implementation of the greedy strategy in rule 1) is shown in [20] (we point out the similarity to [21]):

At each time, we pick the vertex v with highest weight within its neighborhood $N_{\mathcal{G}}(v)$. That is, $v \in \mathcal{S}$ if there is no vertex $u \in N_{\mathcal{G}}(v)$ with a higher weight at a certain time step. So, if $w(v) > w(u) \quad \forall u \in N_{\mathcal{G}}(v)$, then $v \in \mathcal{S}$. For those vertices v with $\exists u \in N_{\mathcal{G}}(v) : w(u) > w(v)$, it depends on the graph structure if $v \in \mathcal{S}$. These vertices v have to wait until they know that $u \in \mathcal{S}$, then $v \notin \mathcal{S}$ or that $u \notin \mathcal{S}$, then $v \in \mathcal{S}$. This can be completed in a finite number of steps.

Theorem 1: The distributed algorithm finds the maximal weighted stable set given by rule 1). The number of steps for the convergence of is bounded above by $2|\mathcal{N}|$.

Proof: The equivalence and convergence of this algorithm is shown in [20]. The complexity in terms of necessary steps is bounded above by $2\alpha(\mathcal{G})$ [20] with $\alpha(\mathcal{G})$ denoting the stability number of $\mathcal{G}$, i.e. the cardinality of the maximum stable set. Naturally, $\alpha(\mathcal{G}) \leq |\mathcal{N}|/2$ since at most half of the nodes can transmit to one single receiver each at the same time. Assuming local information about weights of neighboring vertices, each vertex has to exchange one single message. While in [20] all neighbors can receive this message in one step, this exchange usually needs two steps for the conflict graph as neighboring vertices are in a 2-hop neighborhood in the connectivity hypergraph $\mathcal{H}$. The necessary number of steps is thus bounded above by $2 \cdot 2\alpha(\mathcal{G}) \leq 2|\mathcal{N}|$. ■

Rule 2) was shown [19] to yield a weight of at least $\sum_{v \in \mathcal{V}} w_v/[d_0(v) + 1]$. This value is usually higher than the weight obtained by rule 1). However, local decisions are more complicated as even vertices in a 3-hop neighborhood of $\mathcal{G}$ can influence the weight of neighbors since weights increase with decreasing degree. Before adding a vertex v to $\mathcal{S}$, we have to be sure that it has maximal weight within its neighborhood at a step. So all weights in this 3-hop region have to be known. The number of decision steps is also bounded by $2\alpha(\mathcal{G})$, but there are differences in terms of message exchange. Each decision has to be spread to the 3-hop region. At least for smaller networks this corresponds to flooding the information in the whole network, as message exchange in a 3-hop neighborhood of $\mathcal{G}$ involves up to 6 hops in $\mathcal{H}$. As shown in Fig. 3, this closer approximation of the MWSS problem results in a better performance than the strategy in rule 1). At least for networks of moderate size for which we can compute the optimal solution it performs not too far below the optimum.

In general, bounds on the performance of imperfect scheduling [22] are rather conservative. If the suboptimality of the scheduler can be bounded by a constant factor, the capacity region is also scaled down by at most this factor. The MWSS problem is well-known to be even hard to approximate [23], so bounds of this type are not applicable in this context. On the other hand, Fig. 3 shows that greedily chosen stable sets result

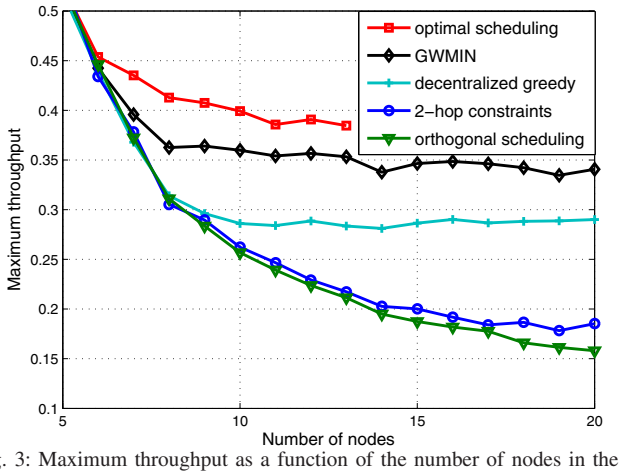

Fig. 3: Maximum throughput as a function of the number of nodes in the network, averaged over 300 random networks.

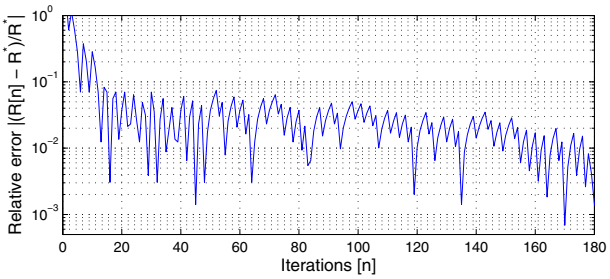

Fig. 4: Convergence of subgradient optimization for an example network with 10 nodes. R^* denotes the optimal throughput, $R[n]$ is the recovered throughput determined by subgradient optimization. Note the logarithmic scale here, so the error is small after a sufficient number of iterations.

in a significantly higher throughput than orthogonal scheduling or the 2-hop constraint. In this setting with constant node density, for both centralized and distributed scheduling approximations the achievable throughput saturates on a certain level depending on the complexity of scheduling. Fig. 4 shows the typical convergence of the subgradient optimization.

VI. Conclusion and Further Work

We presented a framework for the joint optimization problem of scheduling and subgraph selection. As opposed to previous work, we investigated distributed and low-complexity scheduling techniques and evaluated their performance by means of simulations, to conclude that they achieve good performance. The essence of our approach is to isolate the combinatorial difficulty of the problem and to apply decentralized scheduling techniques which perform empirically well. The distributed nature of our approach scales well and makes it attractive for larger networks. It is a further step towards an efficient application of network coding in volatile wireless environments. From a practical point of view, a further challenge is to reduce the required number of steps for convergence while maintaining a distributed realization.

Acknowledgment

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under subcontract No. 18870740-37362-C issued by Stanford University and by the European Commission through NEWCOM++.

References

[1] T. Ho, M. Médard, R. Kötter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Trans. on*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[2] D. S. Lun, N. Ratnakar, M. Médard, R. Kötter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Trans. Netw.*, vol. 14, no. SI, pp. 2608–2623, 2006.

[3] D. Traskov, M. Heindlmaier, M. Médard, R. Kötter, and D. Lun, "Scheduling for network coded multicast: A conflict graph formulation," in *4th IEEE Workshop on Broadband Wireless Access*, New Orleans, LA, USA, 11 2008.

[4] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2003, pp. 66–80.

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

[6] J. Sundararajan, M. Médard, M. Kim, A. Eryilmaz, D. Shah, and R. Kötter, "Network coding in a multicast switch," *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 1145–1153, May 2007.

[7] J. Sundararajan, M. Médard, R. Kötter, and E. Erez, "A systematic approach to network coding problems using conflict graphs," *invited paper, ITA workshop, UCSD*, February 2006.

[8] C. Caramanis, M. Rosenblum, M. Goemans, and V. Tarokh, "Scheduling algorithms for providing flexible, rate-based, quality of service guarantees for packet-switching in banyan networks," *CISS 2004 Princeton, NJ*, 2004.

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

[10] Y. Wu, P. A. Chou, Q. Zhang, K. Jain, and W. Zhu, "Network planning in wireless ad-hoc networks: A cross-layer approach," *IEEE Journal on Selected Areas in Communications*, vol. 23, january 2005.

[11] Y. Sagduyu and A. Ephremides, "On joint mac and network coding in wireless ad hoc networks," *Information Theory, IEEE Transactions on*, vol. 53, no. 10, pp. 3697–3713, Oct. 2007.

[12] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multicast in multi-hop wireless networks," in *WICON '05: Proceedings of the First International Conference on Wireless Internet*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 47–54.

[13] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York: W. H. Freeman, 1979.

[14] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation*. Old Tappan, NJ (USA): Prentice-Hall, Inc., 1989.

[15] D. Bertsekas and R. Gallager, *Data networks (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[16] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming, Theory and Algorithms*. New Jersey: John Wiley and Sons, 2006.

[17] 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, Nov. 1999.

[18] M. Heindlmaier, "Scheduling for network coded multicast," Master's thesis, Institute for Communications Engineering, TUM, 2008.

[19] 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.

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

[21] K. Jung and D. Shah, "Low delay scheduling in wireless network," *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pp. 1396–1400, June 2007.

[22] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, pp. 1804–1814 vol. 3, March 2005.

[23] J. Hastad, "Clique is hard to approximate within $n^{1-\epsilon}$," in *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1996, p. 627.