Chapter 12
Minimum Cost Subgraph Algorithms for Static and Dynamic Multicasts with Network Coding

Fang Zhao, Muriel Médard, Desmond Lun, and Asuman Ozdaglar

12.1 Introduction

Network coding, introduced by Ahlswede et al. in their pioneering work [1], has generated considerable research interest in recent years, and numerous subsequent papers, e.g., [2–6], have built upon this concept. One of the main advantages of network coding over traditional routed networks is in the area of multicast, where common information is transmitted from a source node to a set of terminal nodes. Ahlswede et al. showed in [1] that network coding can achieve the maximum multicast rate, which is not achievable by routing alone. When coding is used to perform multicast, the problem of establishing minimum cost multicast connection is equivalent to two effectively decoupled problems: one of determining the subgraph to code over and the other of determining the code to use over that subgraph. The latter problem has been studied extensively in [5, 7–9], and a variety of methods have been proposed, which include employing simple random linear coding at every node. Such random linear coding schemes are completely decentralized, requiring no coordination between nodes, and can operate under dynamic conditions [10]. These papers, however, all assume the availability of dedicated network resources.

In this chapter, we focus on the former problem, which is to find the min-cost subgraph that allows the given multicast connection to be established (with appropriate coding) over coded packet networks. This problem has been studied in [11, 12]. The analogous problem for routed network is the Steiner tree problem, which is known

Fang Zhao, Muriel Médard, and Asuman Ozdaglar
Massachusetts Institute of Technology, 77 Mass Avenue, Cambridge, MA 02139, USA
e-mail: {zhaof, medard, asuman}@mit.edu

Desmond Lun
The Broad Institute of MIT and Harvard, 7 Cambridge Center, Cambridge, MA 02142, USA
e-mail: dlun@broad.mit.edu

V. Tarokh (ed.), *New Directions in Wireless Communications Research*,
DOI 10.1007/978-1-4419-0673-1_12,

to be NP complete [13,14]. When coding is allowed, the min-cost subgraph problem can be formulated as a linear programming (LP) problem, and in this chapter, we examine algorithms to solve it for both static and dynamic multicasts.

Min-cost Subgraph for Static Multicasts

By static multicast, we refer to the case where a connection is set up for the user of a multicast group whose membership stays constant throughout the connection duration. The network topology, on the other hand, is not necessarily static. Lun et al. showed in [12] that the min-cost subgraph problem can be solved in a decentralized manner by using the dual subgradient method. Since subgraph optimization and coding can be decoupled in the multicast problem with network coding, and both of them can be done in a decentralized manner [10, 12], we have a completely decentralized system for coded multicast. There is no coordination between nodes, each node only knows the costs of its incoming and outgoing links, and communicates with its neighbors. In Section 12.3, we give an overview of this dual subgradient method, and study its convergence performance both theoretically and numerically.

There has been much work on using subgradient methods to solve the Lagrangian dual of a convex constraint optimization problem. The convergence behavior of the subgradient method used on the dual problem is well understood under various step size rules. However, in practice, the main interest is in solving the primal problem, and recovering, from the dual iterations, feasible or near-feasible primal solutions that converge to the optimal solution. There are special cases where the primal solutions computed as a by-product of the dual iterations are feasible and do converge, such as in [15], but this is not the case in general. There are only a few papers studying the recovery of primal solutions from the dual iterations, for example [16–18]. Recently, Nedić and Ozdaglar also looked at the convergence rate of the primal solutions in [19].

In Section 12.3, we present two slightly different formulations of the min-cost subgraph problem for network coding, both of which have the same optimal solutions. These two formulations give rise to two different distributed algorithms. One of them gives us a theoretical bound on the convergence rate of the primal solution; however, its intermediate primal solutions are not always feasible. The second one, on the other hand, produces a feasible subgraph after each iteration, which allows us to start the multicast with minimum delay. We would like to point out that this is possible due to the special structure of the network coding problem, and it is not true in general for dual subgradient method. Thus, coding is central to this formulation of the problem, even though it is not appearing explicitly. More details on this are presented in Section 12.3.1.

We also introduce heuristics to improve the convergence performance of our algorithm, and through simulations, we show that the algorithm produces significant reduction in multicast energy as compared to the centralized routing algorithm just after a few iterations, and it converges to the optimal solution quickly.

One of the challenges of wireless networks, such as ad hoc networks and sensor networks, is variability of the network topology. Topology changes can be caused by mobility of users, sleeping or waking up of nodes, or the shadowing effect due

to moving obstacles. Our algorithm is also put to test in a dynamic wireless network model, and we show that the subgradient method is robust to topology changes, and nodes are able to adjust their transmission power levels to move smoothly and quickly to a new optimal subgraph in a distributed manner.

Min-cost Subgraph for Dynamic Multicasts

In applications such as real-time video distribution and teleconferencing, users can join or leave the multicast group at any time during the session. In such cases, we need to adjust the multicast subgraph to cater for the needs of this dynamic group. Lun et al. gave a dynamic programming formulation of this problem in [20], which aims to deliver continuous service to the users. However, link and code rearrangement, which are defined later, can still occur under their formulation.

In the context of traditional routing networks, this problem corresponds to the dynamic Steiner tree (DST) problem [21]. In DST, it is important to limit the number of rearrangements as a connection evolves, because rearranging a large multipoint connection may be time-consuming and may require significant use of network resources in the form of CPU time. In addition, rearrangement of a connection may result in the blocking of some parts of the connection as rearrangement proceeds. Therefore, the DST problem comes in two flavors [21, 22]. One is the nonrearrangeable version, in which rearrangement of existing routes is not allowed. In the other version, rearrangement is allowed, but the cost of rearrangements is taken into consideration.

The situation is similar in networks with coding. When the membership of the multicast group changes, we want to minimize the disturbance to existing users in the group by limiting both *link rearrangements* and *code rearrangements*. A *link rearrangement* occurs when some links in the current multicast subgraph are removed causing alternate paths to be used to serve existing users (see Fig. 12.1(a) for an example). Like in the routing networks, owing to the change in the physical connection, this kind of rearrangement causes disruptions to the continuous service to the multicast group. The second kind of rearrangement, which we call *code rearrangement*, is more subtle. Code rearrangement occurs when new incoming links are added to existing nodes in the multicast subgraph. Figure 12.1(b) shows an example for code rearrangements. Since random coding is used by the intermediate nodes in the subgraph to perform network coding, when a node has an additional incoming link, it has to generate a new set of random parameters to mix the incoming streams. All receivers downstream, therefore, have to use these new parameters and recompute the inversion matrix to decode the data streams. This scenario does not involve any physical switching of paths for the existing terminals, but it still causes a minor disruption to the continuous service due to this reprocessing of network coding parameters. Note that the disruptions caused by code rearrangements are generally smaller than that caused by link rearrangements.

In Section 12.4 we present algorithms that adapt to the changing demand of the multicast group, and at the same time, minimize disturbances to existing users.We also compare their performances through simulation.

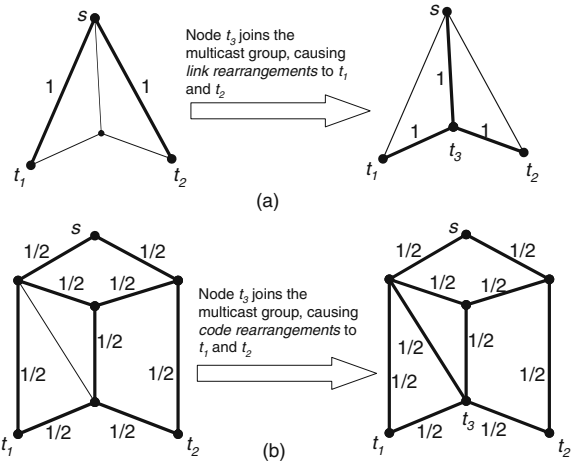

Fig. 12.1 (a) Example of an online step that causes link rearrangements to existing users and (b) example of an online step that causes code rearrangements to existing users. The multicast rate from source node s to terminal nodes $\{t_1, t_2, t_3\}$ is 1. The *thick lines* indicate links used in the multicast and the numbers against them indicate the rate of flow on them

12.2 Problem Formulation

In this section, we present the LP formulation of the min-cost subgraph problem in both wireline and wireless networks. We also derive the Lagrangian dual of these LP problems, which will be used in the distributed algorithms presented in Section 12.3.

12.2.1 Wireline Networks

We look at the problem of single multicast in wireline networks and model the network with a directed graph $G = (N, A)$, where N is the set of nodes and A is the set of links in the network. Each link $(i, j) \in A$ is associated with a non-negative number a_{ij}, which is the cost per unit flow on this link. We assume that the total cost of using a link is proportional to the flow, z_{ij}, on it. For the multicast, suppose we have a source node $s \in N$ producing data at a positive rate R that it wishes to transmit to a non-empty set of terminal nodes T in N.

It is shown in [1] that a subgraph z is capable of supporting a multicast connection of rate R from source s to T if and only if the min-cut from s to any $t \in T$ is greater than or equal to R. Hence, the problem of finding the min-cost subgraph can be formulated into the following LP problem [6]:

$$\begin{aligned}
&\text{minimize } f(z) = \sum_{(i,j)\in A} a_{ij} z_{ij} \\
&\text{subject to } z_{ij} \geq x_{ij}^{(t)} && \forall (i,j) \in A, t \in T, \\
&\sum_{\{j|(i,j)\in A\}} x_{ij}^{(t)} - \sum_{\{j|(j,i)\in A\}} x_{ji}^{(t)} = \delta_i^{(t)} && \forall i \in N, t \in T, \\
&x_{ij}^{(t)} \geq 0, && \forall (i,j) \in A, t \in T,
\end{aligned} \tag{12.1}$$

where $x_{ij}^{(t)}$ corresponds to the virtual flow on link (i, j) for terminal t, z_{ij} is the actual flow on link (i, j) in the multicast subgraph, and

$$\delta_i^{(t)} = \begin{cases} R & \text{if } i = s, \\ -R & \text{if } i = t, \\ 0 & \text{otherwise.} \end{cases}$$

Although the decision variables, z_{ij}, are unbounded in the above formulation, it is easy to see that any optimal solution of (12.1), z^*, is bounded, i.e.,

$$0 \leq z_{ij}^* \leq b_{ij} \quad \forall (i,j) \in A, \tag{12.2}$$

for any $b_{ij} > R$. Thus, including the additional constraint (12.2) in (12.1) would not change the optimal solution set. However, it will affect its Lagrangian dual, and consequently, the algorithm for solving this problem. We will see later in Section 12.3.2 that this additional constraint can help us derive a theoretical bound on the convergence rate of our distributed algorithm.

To derive the Lagrangian dual problem for (12.1), we assign dual variable $p_{ij}^{(t)}$ to the constraint $z_{ij} \geq x_{ij}^{(t)}$ and leave the rest of the primal constraints in the dual objective function. In this way, z_{ij}'s do not appear in the dual problem, and this special structure of the network coding problem allows us to have a feasible primal solution after each dual iteration, as we will see in Section 12.3.1. The dual problem for (12.1) is given by

$$\begin{aligned}
&\text{maximize } q(p) = \sum_{t\in T} q^{(t)}(p^{(t)}) \\
&\text{subject to } \sum_{t\in T} p_{ij}^{(t)} = a_{ij} && \forall (i,j) \in A, \\
&p_{ij}^{(t)} \geq 0 && \forall (i,j) \in A, t \in T,
\end{aligned} \tag{12.3}$$

where

$$q^{(t)}(p^{(t)}) = \min_{x^{(t)} \in F_x^{(t)}} \sum_{(i,j)\in A} p_{ij}^{(t)} x_{ij}^{(t)} \quad \forall t \in T, \tag{12.4}$$

and $F_x^{(t)}$ is the bounded polyhedron of points $x^{(t)}$ satisfying the conservation of flow constraints

$$\sum_{\{j|(i,j)\in A\}} x_{ij}^{(t)} - \sum_{\{j|(j,i)\in A\}} x_{ji}^{(t)} = \delta_i^{(t)} \ \forall i \in N,$$
$$x_{ij}^{(t)} \geq 0 \qquad \forall (i,j) \in A.$$

Note that subproblem (12.4) is a standard shortest path problem with link costs $p_{ij}^{(t)}$, which can be solved using a multitude of distributed algorithms (e.g., distributed Bellman–Ford).

When constraint (12.2) is included in the primal problem, the dual problem becomes

$$\begin{aligned} &\text{maximize } q(p) = \sum_{t\in T} q^{(t)}(p^{(t)}) + \sum_{(i,j)\in A} r_{ij}(p_{ij}) \\ &\text{subject to } p_{ij}^{(t)} \geq 0 \qquad \forall (i,j) \in A, t \in T, \end{aligned} \tag{12.5}$$

where

$$r_{ij}(p_{ij}) = \min_{z\in F_z}(a_{ij} - \sum_{t\in T} p_{ij}^{(t)} z_{ij} \qquad \forall (i,j) \in A,$$

and F_z is the bounded region of z given by

$$0 \leq z_{ij} \leq b_{ij} \qquad \forall (i,j) \in A.$$

12.2.2 Wireless Networks

Under this model, we consider wireless networks where nodes are placed randomly within a 10×10 square with a radius of connectivity r. The energy required to transmit at unit rate to a distance d is taken to be d^2. Let f_{ij} be the cost function of link (i, j), and in our model, $f_{ij}(z_{ij}) = a_{ij} z_{ij}$ where $a_{ij} = d_{ij}^2$ is the energy required to send at unit rate over this link and z_{ij} is the rate of flow on this link. We justify this assumption in the cases where energy is the most significant constraint, but there are, for example, sufficient time or frequency slots to guarantee that no two transmissions ever interfere. This model is discussed in more depth in [6].

We consider wireless networks in which antennas are omnidirectional. When we transmit from node i to node j, we get transmission to all nodes whose distance from i is less than that from i to j "for free" — a phenomenon referred to as the "wireless multicast advantage" in [23]. If we impose an ordering $\preceq$ on the set of outgoing links from i, such that $(i, k) \preceq (i, j)$ if and only if $a_{ik} \leq a_{ij}$, we can then assume that we obtain a lossless broadcast link of unit rate from node i to all nodes k such that $(i, k) \preceq (i, j)$ for cost a_{ij}. Consider the example shown in Fig. 12.2, where there are three nodes within distance r from node i. If node i transmits with power a_{ij_3} to node j_3, the two nearer nodes, j_1 and j_2, also receive this information without additional cost. Thus, the situation here is quite different from the wireline case. Instead of picking links to transmit on in the wireline networks, the nodes in wireless networks pick power levels to transmit with, and this in turn determines their radius of coverage.

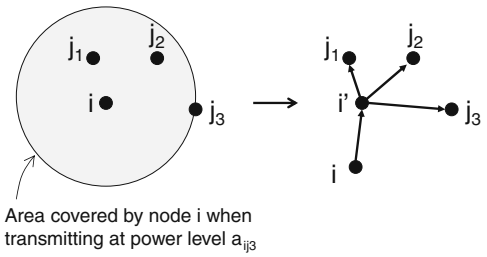

Fig. 12.2 The "wireless multicast advantage" associated with omnidirectional antennas. The three destinations j_1, j_2, and j_3 can all be reached at the same time with cost a_{ij_3}, and this is equivalent to having three unit capacity links from i to j_1, j_2, and j_3. Here, we also included a virtual unit-capacity link (i, i') to impose the constraint that information transmitted on the three links must be the same

Similar to the wireline case, in the wireless network, the min-cost subgraph that can be used to perform the multicast with network coding is given by the following linear optimization problem:

$$\begin{aligned}
&\text{minimize } f(z) = \sum_{(i,j)\in A} a_{ij} z_{ij} \\
&\text{subject to } \sum_{\{k|(i,k)\in A,(i,k)\succeq(i,j)\}} (z_{ik} - x_{ik}^{(t)}) \geq 0 \quad \forall (i,j) \in A', t \in T, \\
&\qquad \sum_{\{j|(i,j)\in A\}} x_{ij}^{(t)} - \sum_{\{j|(j,i)\in A\}} x_{ji}^{(t)} = \sigma_i^{(t)} \ \forall i \in N, t \in T, \\
&\qquad x_{ij}^{(t)} \geq 0 \qquad\qquad \forall (i,j) \in A, t \in T,
\end{aligned} \tag{12.6}$$

Here, A' is a subset of A with the property that the constraint

$$\sum_{\{k|(i,k)\in A,(i,k)\succeq(i,j)\}} (z_{ik} - x_{ik}^{(t)}) \geq 0$$

is unique for all $(i, j) \in A'$.

The subgraph optimization scheme also uses the Lagrangian dual of (12.6) given below:

$$\begin{aligned}
&\text{maximize } q(p) = \sum_{t\in T} q^{(t)}(p^{(t)}) \\
&\text{subject to } \sum_{\{k|(i,k)\in A',(i,k)\preceq(i,j)\}} \sum_{t\in T} p_{ik}^{(t)} = a_{ij} \ \forall (i,j) \in A, \\
&\qquad p_{ij}^{(t)} \geq 0 \qquad\qquad \forall (i,j) \in A', t \in T,
\end{aligned} \tag{12.7}$$

where

$$q^{(t)}(p^{(t)}) = \min_{x^{(t)}\in F_x^{(t)}} \sum_{(i,j)\in A} \left(\sum_{\{k|(i,k)\in A',(i,k)\preceq(i,j)\}} p_{ik}^{(t)} \right) x_{ij}^{(t)} \tag{12.8}$$

and $F^{(t)}$ is the bounded polyhedron of points $x^{(t)}$ satisfying the conservation of flow constraints. Again, z_{ij}'s do not appear in the dual problem, so we can recover a feasible primal solution after each dual iteration, which is not possible for general subgradient methods.

To simplify the constraints in the dual problem (12.7), we can sort the outgoing links from node i in A' according to their costs. Note that in A', no two outgoing links from a node i are of the same cost. Consider the example in Fig. 12.2 where there are three outgoing links from i with $(i, j_1) \prec (i, j_2) \prec (i, j_3)$, the equality constraints in (12.7) with respect to these links become

$$\begin{aligned}
&\sum_{t\in T} p_{ij_1}^{(t)} = a_{ij_1},\\
&\sum_{t\in T} p_{ij_1}^{(t)} + \sum_{t\in T} p_{ij_2}^{(t)} = a_{ij_2},\\
&\sum_{t\in T} p_{ij_1}^{(t)} + \sum_{t\in T} p_{ij_2}^{(t)} + \sum_{t\in T} p_{ij_3}^{(t)} = a_{ij_3}.
\end{aligned}$$

These are equivalent to

$$\begin{aligned}
&\sum_{t\in T} p_{ij_1}^{(t)} = a_{ij_1},\\
&\sum_{t\in T} p_{ij_2}^{(t)} = a_{ij_2} - a_{ij_1},\\
&\sum_{t\in T} p_{ij_3}^{(t)} = a_{ij_3} - a_{ij_2}.
\end{aligned}$$

Therefore, if we define

$$s_{ij} = a_{ij} - \max_{\{k|(i,k)\in A', (i,k)\prec(i,j)\}} a_{ik}, \tag{12.9}$$

the dual problem (12.7) can be simplified to

$$\begin{aligned}
&\text{maximize} \sum_{t\in T} q^{(t)}(p^{(t)}) \qquad\qquad (12.10)\\
&\text{subject to} \sum_{t\in T} p_{ij}^{(t)} = s_{ij} \ \forall (i,j) \in A,\\
&\qquad\qquad p_{ij}^{(t)} \geq 0 \qquad \forall (i,j) \in A', t \in T.
\end{aligned}$$

12.3 Decentralized Min-cost Subgraph Algorithms for Static Multicast

We focus on static multicasts in this section. Section 12.3.1 gives an overview of the dual subgradient method for decentralized subgraph optimization. The convergence rate of this method is analyzed in Section 12.3.2. Various heuristics to improve the convergence performance of the canonical algorithm in both static and dynamic

wireless networks are presented in Section 12.3.3, and Section 12.3.4 gives some numerical results.

12.3.1 Subgradient Method for Decentralized Subgraph Optimization

The subgraph optimization scheme in [6] tries to converge to the optimal primal solution by using a subgradient method on the dual problem. This algorithm is completely decentralized and each node has to know only the cost of its incoming and outgoing links, and exchange information with neighboring nodes. We first give an overview of the algorithm under the wireline network model in Section 12.3.1.1. Section 12.3.1.2 describes the extension of this algorithm to the wireless case.

12.3.1.1 Subgradient Method in Wireline Networks

In the following, we describe the distributed algorithms for solving (12.1) with and without constraint (12.2). We refer to the algorithm that solves problem (12.1) and its dual (12.3) as *Algorithm A*, and the algorithm for solving the primal with constraint (12.2) and dual (12.5) as *Algorithm B*. Most of the discussions and simulations in Section 12.3 are based on Algorithm A, since it has better convergence performance in practical settings. However, in Section 12.3.2, we use Algorithm B to derive a theoretical bound on the convergence rate of the primal solutions, which is not available for Algorithm A.

Algorithm A

1. **Initialize $p[0]$** — Before the first iteration, each node initializes $p[0]$.
2. **Compute $x[n]$** — In the nth iteration, use $p[n]$ as link costs, and run a distributed shortest path algorithm to determine $x[n]$.
3. **Update $p[n+1]$** — Update $p[n+1]$ using subgradient obtained through $x[n]$ values:

$$g_{ij}^{(t)}[n] = x_{ij}^{(t)}[n],$$

$$p[n+1] := \left[p[n] + \theta[n] g^{(t)}[n] \right]_P^+,$$

where $g[n]$ is the subgradient for $p[n]$, $\theta[n]$ is the step size for the nth iteration, and $[\cdot]_P^+$ denotes the projection onto the constraint set P in (12.3). This projection can be done in a distributed manner, and specifically $p_{ij}^{(t)}[n+1]$ is given by

$$p_{ij}^{(t)}[n+1] = \max\left(0,\, p_{ij}^{(t)}[n] + \theta[n] x_{ij}^{(t)}[n] + d_{ij}[n]\right), \tag{12.11}$$

where $d_{ij}[n] \leq 0$ is a number computed based on the $p[n]$, $x[n]$, and $\theta[n]$ values [6].

4. **Recover $\tilde{x}[n]$** — At the end of each iteration, nodes recover a primal solution, $\tilde{x}[n]$, based on the dual computations. Let $\{\mu_l[n]\}_{l=1,\ldots,n}$ be a sequence of convex combination weights for each non-negative integer n, i.e., $\sum_{l=1}^{n} \mu_l[n] = 1$ and $\mu_l[n] \geq 0$ for all $l = 1, \ldots, n$. Further, let us define

$$\gamma_{ln} = \frac{\mu_l[n]}{\theta[l]}, \qquad l = 1, \ldots, n, \quad n = 0, 1, \ldots,$$

and

$$\Delta\gamma_n^{\max} = \max_{l=2,\ldots,n} \{\gamma_{ln} - \gamma_{(l-1)n}\}.$$

According to [16], if the step sizes $\{\theta[n]\}$ and the convex combination weights $\{\mu_l[n]\}$ are chosen such that

a. $\gamma_{ln} \geq \gamma_{(l-1)n}$ for all $l = 2, \ldots, n$, and $n = 0, 1, \ldots$,

b. $\Delta\gamma_n^{\max} \to 0$ as $n \to \infty$, and

c. $\gamma_{1n} \to 0$ as $n \to \infty$ and $\gamma_{nn} \leq \delta$ for all $n = 0, 1, \ldots$, for some $\delta > 0$,

then we obtain an optimal solution to the primal problem (12.1) from any accumulation point of the sequence of primal iterates $\{\tilde{x}[n]\}$ given by

$$\tilde{x}_{ij}^{(t)}[n] = \sum_{l=1}^{n} \mu_l[n] x_{ij}^{(t)}[l], \qquad n = 0, 1, \ldots.$$

An example of a set of parameters that satisfy the above conditions are $\theta[n] = n^{-\alpha}$ for $n = 0, 1, 2, \ldots$ where $0 < \alpha < 1$, and $\mu_l[n] = 1/n$ for $n = 1, 2, 3, \ldots$ and $l = 1, \ldots, n$.

5. **Determine $\tilde{z}[n]$** — Each node computes the $\tilde{z}_{ij}[n]$ values from the $\tilde{x}_{ij}^{(t)}[n]$ values. In order to minimize the cost, $\tilde{z}_{ij}[n] = \max_{t \in T} \tilde{x}_{ij}^{(t)}[n]$.
6. **Repeat** — Steps 2–5 are repeated until the primal solution has converged.

For details of this algorithm and related proofs, refer to [6].

Since the intermediate $\{\tilde{z}[n], \tilde{x}[n]\}$ values after each iteration are always feasible solutions to the primal problem, we do not have to wait till the primal solution converges to start the multicast. Instead, the multicast can be started after the first iteration, and we can shift the flows gradually through the iterations to operate on a more cost-effective subgraph. Note that, in general, this is not true for dual subgradient methods, and it works out here due to the unique structure of the network coding problem. Specifically, the flow variables, z_{ij}, are not involved in the flow conservation constraints, and they do not appear in the dual iterations. This allows us to pick feasible z-values after each dual iteration based on a set of feasible virtual flows x.

If the boundedness constraint (12.2) is included in the primal problem, we have Algorithm B for solving this new problem and its dual (12.5).

Algorithm B

1. **Initialize** $p[0]$.
2. **Compute $x[n]$ and $z[n]$** — Computation of $x[n]$ is the same as that in Algorithm A. For $z[n]$, we have

$$z_{ij}[n] = \begin{cases} 0 & \text{if } \sum_{t \in T} p_{ij}^{(t)} \leq a_{ij}, \\ b_{ij} & \text{if } \sum_{t \in T} p_{ij}^{(t)} \geq a_{ij}. \end{cases}$$

3. **Update** $p[n+1]$ — Update $p[n+1]$ using subgradient obtained through $x[n]$ and $z[n]$ values:

$$g_{ij}^{(t)}[n] = x_{ij}^{(t)}[n] - z_{ij}[n],$$

$$p_{ij}^{(t)}[n+1] = \max\left(0,\, p[n] + \theta[n] g_{ij}^{(t)}[n]\right).$$

4. **Recover $\tilde{z}[n]$** — Recovery of the primal solution $\tilde{z}[n]$ is done by taking a convex combination of all past $z[n]$ values, similar to the recovery of $\tilde{x}[n]$ in Algorithm A:

$$\tilde{z}_{ij}^{(t)}[n] = \sum_{l=1}^{n} \mu_l[n] z_{ij}^{(t)}[l], \qquad n = 0, 1, \ldots.$$

5. **Repeat** — Steps 2–4 are repeated until the primal solution has converged.

As we shall see in Section 12.3.2, Algorithm B provides us a theoretical bound on the primal convergence rate of the min-cost subgraph problem, which is not available for Algorithm A. However, a major drawback of Algorithm B compared to Algorithm A is that the $\tilde{z}[n]$ values are not always feasible; therefore, we cannot start the multicast right away as in the case of Algorithm A. This is very undesirable in practice and is one of the main reasons why we only focus on Algorithm A in our simulations.

12.3.1.2 Subgradient Method in Wireless Networks

The main steps in the distributed min-cost subgraph algorithm for wireless networks are the same as that in Algorithm A of the wireline case, except for steps 3 and 5, in which some modifications are required. The details of the changes are highlighted below.

In step 3, when updating $p[n+1]$, the subgradient for $p_{ij}^{(t)}[n]$ in the wireless case is given by

$$g_{ij}^{(t)}[n] = \sum_{\{k \mid (i,k) \in A, (i,k) \succeq (i,j)\}} x_{ik}^{(t)}[n],$$

and again, $p_{ij}[n+1]$ is the Euclidean projection of $p_{ij}[n] + \theta[n] g_{ij}[n]$ onto the feasible set P_{ij}.

In step 5, we compute $\tilde{z}[n]$ based on the recovered primal solution $\tilde{x}[n]$. Recall that in the primal problem (12.6) we have the constraints

$$\sum_{\{k|(i,k)\in A,(i,k)\succeq(i,j)\}} (\tilde{z}_{ik} - x_{ik}^{(t)}) \geq 0 \ \forall (i,j) \in A', t \in T. \tag{12.12}$$

Assume that the sorted list of outgoing links from node i in A' based on their costs is $\{(i, j_1), \ldots, (i, j_k)\}$, and start from the most expensive links (i, j_k), the above constraint becomes

$$\tilde{z}_{ij_k} - \tilde{x}_{ij_k}^{(t)} \geq 0 \ \forall t \in T.$$

To minimize total cost, the optimal $\tilde{z}_{ij_k}$ value should be $\max_{t\in T} \tilde{x}_{ij_k}^{(t)}$. In cases where more than one outgoing links are of the same cost, we just need to make sure that the sum of the flows on these links satisfy constraint (12.12). The distribution of the total flow among these links can be done randomly without affecting the total cost. Once $\tilde{z}_{ij_k}$ value is determined, we can move on to the second most expensive link (i, j_{k-1}), whose constraint now becomes

$$\left(\tilde{z}_{ij_{k-1}} - \tilde{x}_{ij_{k-1}}^{(t)}\right) + \left(\tilde{z}_{ij_k} - \tilde{x}_{ij_k}^{(t)}\right) \geq 0 \ \forall t \in T,$$

and we have $\tilde{z}_{ij_{k-1}} = \max_{t\in T}(\tilde{x}_{ij_{k-1}}^{(t)} + \tilde{x}_{ij_k}^{(t)}) - z_{ij_k}$. By repeating the above process, we can obtain the optimal primal solution $\tilde{z}$ from $\tilde{x}$.

12.3.2 Convergence Rate Analysis

In this section, we study the convergence rates of our dual subgradient method presented in Section 12.3.1 in both the primal and the dual spaces. For clarity of presentation, we use the wireline model in this section, as its notations are much simpler than the wireless one. All results here can be easily extended to the wireless case.

12.3.2.1 Convergence Rate for the Dual Problem

The analysis and results in this section apply to both Algorithm A and Algorithm B. Here, we just present the analysis for Algorithm A, as the extension to Algorithm B is fairly straightforward.

With properly chosen step sizes, the standard subgradient method proposed in Section 12.3.1 converges to dual optimal solutions eventually [24], but it is difficult to analyze the convergence rate of the standard method. To this end, we consider the incremental subgradient method studied in [25]. The incremental subgradient method can be used here because the objective function in (12.3) is the sum of $|T|$ convex component functions, and the constraint set is non-empty, closed, and convex (see Chapter 2 of [25]). At each iteration, p is changed incrementally through

a sequence of $|T|$ steps. Each step is a subgradient iteration for a single component function $q^{(t)}$. Thus, an iteration can be viewed as a cycle of $|T|$ subiterations. Denote the terminal nodes by $\{1, 2, \ldots, N_T\}$, where $N_T = |T|$. The vector $p[n+1]$ is obtained from $p[n]$ as follows:

$$\begin{aligned} \psi_0[n] &:= p[n], \\ \psi_i[n] &:= \left[\psi_{i-1}[n] + \theta[n] g^{(i)}[n]\right]_P^+, \\ p[n+1] &:= \psi_{N_T}[n]. \end{aligned}$$

We first prove two propositions that are useful for the convergence rate analysis.

Proposition 1 *Problem (12.3) satisfies the subgradient boundedness property, which means there exists a positive scalar C such that*

$$\begin{aligned} ||g|| \le C,\ \forall g \in \partial q^{(t)}(p[n]) \cup \partial q^{(t)}(\psi_{i-1,n}), \\ \forall i = 1, \ldots, N_T \qquad \forall n. \end{aligned}$$

Proof. This is true because $q^{(t)}$ is the pointwise minimum of a finite number of affine functions, and in this case, for every p, the set of subgradients $\partial q^{(t)}(p)$ is the convex hull of a finite number of vectors. Thus, the subgradients are bounded. □

Proposition 2 *Let the optimal solution set be P^*, there exists a positive scalar μ such that*

$$q^* - q(p) \ge \mu(\text{dist}(p, P^*))^2 \qquad \forall p \in P.$$

Proof. Problem (12.3) can be reformulated into a linear programming problem as follows:

$$\begin{aligned} \text{maximize } & q'(v) = \sum_{t\in T}\sum_{i\in N} r_i^{(t)}\delta_i^{(t)} = R\sum_{t\in T}\left(r_s^{(t)} - r_t^{(t)}\right) \\ \text{subject to } & r_i^{(t)} - r_j^{(t)} \le p_{ij}^{(t)} \qquad \forall (i,j) \in A,\ t \in T, \\ & \sum_{t\in T} p_{ij}^{(t)} = a_{ij} \qquad \forall (i,j) \in A, \\ & p_{ij}^{(t)} \ge 0 \qquad\qquad \forall (i,j) \in A, t \in T. \end{aligned} \tag{12.13}$$

The decision vector, v, is a concatenation of vectors p and r, and we denote the feasible set by V. For any feasible $p \in P$ from (12.3), there is a corresponding v in (12.13) with the same p-component and $q'(v) = q(p)$. Furthermore, for any feasible $v \in V$, we can extract a p vector from it that gives the same total cost in (12.3). Therefore, the two formulations (12.3) and (12.13) have the same optimal values, i.e., $q^* = q'^*$.

Since the set of solutions for a linear programming problem is a set of weak sharp minima [26], there exists a positive α such that

$$q'^* - q'(v) \ge \alpha(\text{dist}(v, V^*)) \qquad \forall v \in V.$$

So for any $p \in P$ in (12.3), we have

$$q^* - q(p) = q'^* - q'(v) \geq \alpha(\text{dist}(v, V^*)) \geq \alpha(\text{dist}(p, P^*)).$$

The last inequality comes from the fact that p/P^* is the projection of v/V^* on P, and the projection operation is non-expansive. Since P is a bounded polyhedron, the distance between any two points in P is bounded, i.e., $\text{dist}(p, p') \leq B$ for all $p, p' \in P$ for some positive B. Therefore,

$$q^* - q(p) \geq \frac{\alpha}{B}(\text{dist}(p, P^*))^2.$$

Let $\mu = \alpha/B$, and the proposition is proved. □

With these propositions, we have the following result for constant step size.

Proposition 3 *For the sequence $\{p[n]\}$ generated by the incremental subgradient method with the step size $\theta[n]$ fixed to some positive constant θ, where $\theta \leq \frac{1}{2\mu}$, we have*

$$(\text{dist}(p[n+1], P^*))^2 \leq (1 - 2\theta\mu)^{n+1}(\text{dist}(p[0], P^*))^2 + \frac{\theta|T|^2C^2}{2\mu} \qquad \forall n. \tag{12.14}$$

Proof. The proof for this proposition follows from Proposition 1.2 and the proof of Proposition 2.3 in [25]. Since the dual problem satisfies Proposition 1 (bounded subgradient), from Lemma 2.1 in [25], we have

$$||p[n+1]-r||^2 \leq ||p[n]-r||^2 - 2\theta(q(r)-q(p[n])) + \theta^2|T|^2C^2 \qquad \forall r \in P, \quad \forall n.$$

Using this relation with $r = p^*$ for any optimal $p^* \in P^*$, we see that

$$||p[n+1]-p^*||^2 \leq ||p[n]-p^*||^2 - 2\theta(q^*-q(p[n])) + \theta^2|T|^2C^2 \qquad \forall r \in P, \quad \forall n,$$

and by taking the minimum over all $p^* \in P^*$, we have (12.15)

$$\begin{aligned}(\text{dist}(p[n+1], P^*))^2 &\leq (\text{dist}(p[n], P^*))^2 - 2\theta(q^* - q(p[n])) + \theta^2|T|^2C^2 \\ &\leq (1 - 2\theta\mu)(\text{dist}(p[n], P^*))^2 + \theta^2|T|^2C^2 \qquad \forall n,\end{aligned} \tag{12.16}$$

where the last inequality comes from Proposition 2. From this relation, by induction, we can see that

$$\begin{aligned}(\text{dist}(p[n+1], P^*))^2 \leq &(1 - 2\theta\mu)^{(n+1)}(\text{dist}(p[0], P^*))^2 \\ &+ \theta^2|T|^2C^2 \sum_{i=0}^{n}(1 - 2\theta\mu)^i \qquad \forall n,\end{aligned}$$

which combined with

$$\sum_{i=0}^{n}(1 - 2\theta\mu)^i \leq \frac{1}{2\theta\mu},$$

yields the desired relation (12.14). □

In summary, we have shown that the convergence rate for the incremental subgradient method on (12.3) is linear for a sufficiently small step size. However, only convergence to a neighborhood of the optimal solution set can be guaranteed, which is typical for constant step size rules. Moreover, our result also highlights the trade-off between the error and the convergence rate constant. The smaller the θ value, the smaller the size of the neighborhood, but on the other hand, we get a slower convergence.

12.3.2.2 Convergence Analysis for the Primal Problem

As mentioned in Section 12.3.1.1, it is advantageous to use Algorithm A in practice, as its primal solution is always feasible through the iterations. Unfortunately, due to the unboundedness of z_{ij} in formulation (12.1), it is very hard to derive its primal convergence rate. Therefore, in this section, we turn our focus to Algorithm B, for which we derive a bound on its convergence rate.

We first prove that our primal problem (12.1) with constraint (12.2) satisfies the Slater condition in Proposition 4, then present the main convergence rate result in Proposition 5.

Proposition 4 The Slater condition *There exists a vector* $\{\bar{z}, \bar{x}\}$, $\{\bar{z}, \bar{x}\} \in F$ *such that*

$$\bar{z}_{ij} > \bar{x}_{ij}^{(t)} \qquad \forall (i, j) \in A, t \in T, \tag{12.17}$$

where $F = \{F_z, F_x\}$ *is the feasible set for the boundedness constraints for* z *and the conservation of flow constraints for* x.

Proof. For the virtual flows, $x_{ij}^{(t)}$, based on the conservation of flow constrains, there exists feasible solutions where $x_{ij}^{(t)} \leq R$ for all $(i, j) \in A$ and $t \in T$. Since the upper bound on z_{ij} is $b_{ij} > R$, we can always find a set of z that is strictly greater than the corresponding x. Therefore, our primal problem satisfies the Slater condition. □

Proposition 5 *Let* $\{\bar{z}, \bar{x}\}$ *be a Slater vector satisfying (12.17), and* C *be the subgradient norm bound in Proposition 1, define*

$$B^* = \frac{2}{\gamma}(f(\bar{z}) - q^*) + \max\left\{||p[0]||, \frac{1}{\gamma}(f(\bar{z}) - q^*) + \frac{\theta C^2}{2\gamma} + \theta C\right\},$$

where $\gamma = \min_{\{i,j\}\in A, t\in T}\left(\bar{z}_{ij}^{(t)} - \bar{x}_{ij}^{(t)}\right)$. *If constant step size* θ *is used in the dual iterations, and simple averaging is used in the primal recovery, i.e.,* $\mu_l[n] = 1/n$ *for* $l = 1, 2, \ldots, n$, *then the primal cost after the nth iteration is bounded by*

$$f^* - \frac{1}{\gamma}[f(\bar{z}) - q^*]\frac{B^*}{n\theta} \leq f(\bar{z}[n]) \leq f^* + \frac{\|p[0]\|^2}{2n\theta} + \frac{\theta C^2}{2}. \tag{12.18}$$

Proof. We first derive the lower bound on $f(\tilde{z}[n])$ (the left-hand side of (12.18)). Recall that in Algorithm B, $g_{ij}^{(t)}[n] = g\left(z_{ij}[n], x_{ij}^{(t)}[n]\right) = x_{ij}^{(t)}[n] - z_{ij}[n]$, and

$$p[n+1] = \max(0,\, p[n] + \theta g[n]) \geq p[n] + \theta g[n],$$

we have

$$\theta g(z[n], x[n]) \leq p[n+1] - p[n] \qquad \forall n \geq 0.$$

Therefore, $\sum_{i=0}^{n-1} \theta g(z[i], x[i]) \leq p[n] - p[0] \leq p[n]$, where the last inequality follows from $p[0] \geq 0$. By the convexity of the function g, it follows that

$$g(\tilde{z}[n], \tilde{x}[n]) \leq \frac{1}{n} \sum_{i=0}^{n-1} g(z[i], x[i]) = \frac{1}{n\theta} \sum_{i=0}^{n-1} \theta g(z[i], x[i]) \leq \frac{p[n]}{n\theta}.$$

Because $p[n] \geq 0$, the positive elements in $g(\tilde{z}[n], \tilde{x}[n])$ satisfy $g(\tilde{z}[n], \tilde{x}[n])^+ \leq p[n]/n\theta$ for all $n \geq 0$. Let the amount of constraint violation of $(\tilde{z}[n], \tilde{x}[n])$ be $||g(\tilde{z}[n], \tilde{x}[n])^+||$, we have

$$\|g(\tilde{z}[n], \tilde{x}[n])^+\| \leq \frac{\|p[n]\|}{n\theta} \qquad \forall n \geq 1. \tag{12.19}$$

Given a dual optimal solution p^*, we have

$$q(p^*) = q^* \leq f(z) + (p^*)' g(z, x)$$

for any $x \in F_x$ and $z \in F_z$. Thus,

$$\begin{aligned} f(\tilde{z}[n]) &= f(\tilde{z}[n]) + (p^*)' g(\tilde{z}[n], \tilde{x}[n]) - (p^*)' g(\tilde{z}[n], \tilde{x}[n]) \\ &\geq q^* - (p^*)' g(\tilde{z}[n], \tilde{x}[n]). \end{aligned} \tag{12.20}$$

Because $p^* \geq 0$ and $g(\tilde{x}[n], \tilde{x}[n])^+ \geq g(\tilde{x}[n], \tilde{x}[n])$, we further have

$$-(p^*)' g(\tilde{z}[n], \tilde{x}[n]) \geq -(p^*)' g(\tilde{z}[n], \tilde{x}[n])^+ \geq -||p^*\| \|g(\tilde{z}[n], \tilde{x}[n])^+\|. \tag{12.21}$$

From (12.19), (12.20), and (12.21), it follows that

$$f(\tilde{z}[n]) \geq q^* - \|p^*\| \frac{\|p[n]\|}{n\theta}. \tag{12.22}$$

Since our primal problem satisfies the Slater condition and the dual iterates have bounded subgradients, from Lemma 1 and 3 in [19], we have

$$\|p^*\| \leq \frac{1}{\gamma}(f(\bar{z}) - q^*) \qquad \text{and} \qquad \|p[n]\| \leq B^* \qquad \forall n \geq 1,$$

where γ and B^* are defined in the above proposition. Substitute these bounds into (12.22), and we have the lower bound on $f(\tilde{z}[n])$

$$f(\tilde{z}[n]) \geq q^* - \frac{1}{\gamma}[f(\bar{z}) - q^*]\frac{B^*}{n\theta}.$$

Next, we derive the upper bound on $f(\tilde{z}[n])$ (right-hand side of (12.18)). By the convexity of $f(z)$ and the definition of $(z[n], x[n])$ as a minimizer of the Lagrangian function $f(z) + p'g(z, x)$ over $x \in F_x$ and $z \in F_z$, we have

$$\begin{aligned} f(\tilde{z}[n]) &\leq \frac{1}{n}\sum_{i=0}^{n-1} f(z[i]) \\ &= \frac{1}{n}\sum_{i=0}^{n-1}\big(f(z[i]) + p[i]'g(z[i], x[i])\big) - \frac{1}{n}\sum_{i=0}^{n-1} p[i]'g(z[i], x[i]) \\ &= \frac{1}{n}\sum_{i=0}^{n-1} q(p[i]) - \frac{1}{n}\sum_{i=0}^{n-1} p[i]'g(z[i], x[i]) \\ &\leq q^* - \frac{1}{n}\sum_{i=0}^{n-1} p[i]'g(z[i], x[i]). \end{aligned} \tag{12.23}$$

Since $p[n+1] = \big[p[n] + \theta g[n]\big]^+$, by using the non-expansive property of projection and the fact that 0 is in the feasible region of the dual problem (12.5), we have

$$\|p[i+1]\|^2 \leq \|p[i]\|^2 + 2\theta p[i]'g[i] + \theta^2\|g[i]\|^2.$$

Since $g[i] = g(z[i], x[i])$, we further obtain

$$-p[i]'g(z[i], x[i]) \leq \frac{\|p[i]\|^2 - \|p[i+1]\|^2 + \theta^2\|g(z[i], x[i])\|^2}{2\theta}, \quad 0 \leq i \leq n-1.$$

By summing over $i = 0, 1, \ldots, n-1$, and combining with (12.23), we have

$$\begin{aligned} f(\tilde{z}[n]) &\leq q^* + \frac{\|p[0]\|^2 - \|p[n]\|^2}{2n\theta} + \frac{\theta}{2n}\sum_{i=0}^{n-1}\|g(z[i], x[i])\|^2 \\ &\leq q^* + \tfrac{\|p[0]\|^2}{2n\theta} + \tfrac{\theta C^2}{2} \qquad \forall n \geq 1. \end{aligned} \tag{12.24}$$

By combining (12.22) and (12.24), we have the desired relation. □

This proposition shows that when using constant step size, the primal solutions converge to a neighborhood of the optimal solution with rate $O(1/n)$. Note that there is a trade-off between the size of the neighborhood and the convergence rate. If we want the primal solution to be close to the optimal one, we need to choose a small step size, but this would make the convergence rate very slow. This is a typical problem with using constant step size, and in our simulations, we avoid this situation by using diminishing step size.

12.3.3 Initialization and Primal Solution Recovery

In order to improve the convergence performance of the subgradient algorithm, we introduce some heuristics for steps 1 and 4 in the algorithm presented in Section 12.3.1.2. Specifically, we propose several methods for initializing the dual vector $p[0]$, and for recovering primal solutions $\{\tilde{x}[n]\}$, for both static and dynamic wireless networks.

12.3.3.1 Static Networks

We start with static networks, where the topology of the network is fixed throughout the multicast. We first introduce a naive way of initializing the dual variables.

- **Averaging method** — The simplest way to generate feasible initial values for the dual variables is to assign $p_{ij}^{(t)} = s_{ij}/N_T$ for all $t \in T$ and all $(i, j) \in A$. This method is useful in static networks since no prior information of the multicast problem is available at the nodes.

  For the recovery of primal solution $\tilde{x}[n]$, we have the following two options.

- **Original primal recovery** — This is the recovery method presented in step 4 in Section 12.3.1 with simple averaging.
- **Modified primal recovery** — Using the original primal recovery method, we observed in simulations that the cost of the multicast starts at a high value, and then converges slowly to the optimal value through iterations. One reason for the slow convergence is that it is recovered by averaging $x[n]$ values from all the iterations. The effect of the first few high-cost iterations takes a large number of iterations later to dilute. A heuristic way to improve the convergence rate is to discard these "bad" primal solutions after some time, and just average over the most recent N_a number of iterations in primal solution recovery.

12.3.3.2 Dynamic Networks

As opposed to the static assumption, many wireless networks have topologies that are dynamic. Whenever a topology change occurs, we need to restart the distributed algorithm, as the subgraph used for multicast before the topology change might have become infeasible. In such cases, all the methods discussed in the previous section for dual variable initialization and primal solution recovery are still applicable. However, since the difference between the optimal solutions to the multicast problem before and after the changes is usually small, we should make use of the solutions $\hat{x}$ and $\hat{p}$ before the topology changes in the new iterations to improve the convergence rate. We propose additional methods to initialize p and update $\tilde{x}$ that make use of this old information.

For dual variable initialization, we present two additional heuristics.

- **Scaling method** — In this method, each node i scans through its set of outgoing links in A'. If a link (i, j) is an existing link in $\hat{A}'$ before the topology change, scale the $\{\hat{p}_{ij}^{(t)}\}$ values so that they satisfy the new dual constraints. Specifically, denoting $\sum_{t\in T} \hat{p}_{ij}^{(t)} = \hat{s}_{ij}$, we assign

$$p_{ij}^{(t)} = \hat{p}_{ij}^{(t)} \times \frac{s_{ij}}{\hat{s}_{ij}}.$$

  On the other hand, if link (i, j) is a new link after the topology change, we simply use

$$p_{ij}^{(t)} = s_{ij}/|T|.$$

- **Projection method** — In this method, we use an intermediate $\tilde{P}$ which is given by

$$\tilde{p}_{ij}^{(t)} = \begin{cases} \hat{p}_{ij}^{(t)} & \text{if } (i, j) \text{ is an old link,} \\ 0 & \text{if } (i, j) \text{ is a new link.} \end{cases}$$

  We can then project this $\tilde{P}$ onto the new feasible region of the dual problem using (12.11) to obtain an initial point P for the decentralized algorithm.

On the primal side, we observe that as long as no removed link was in the multicast subgraph used before the topology change, the old $\{\hat{x}[n]\}$ values from the previous iterations are still valid under the new topology. Thus, they can be used in the recovery of the current primal optimal solution. Based on this observation, we propose the following heuristics for primal solution recovery.

- **Look-back primal recovery** — When a topology change occurs, each node checks if any of its links used in the multicast is removed owing to topology change. If yes, it sends out a signal to all nodes, and $\{\tilde{x}[n]\}$ is computed based only on the new $\{x[n]\}$ values as in the original primal recovery method above. On the other hand, if no link is removed, the averaging is done over N_a iterations before and after the topology change. The assumption that nodes can be informed of the removal of an active link within one iteration is reasonable, since, in each iteration, distributed Bellman–Ford is used to compute $x[n]$ and sending such a signal to all nodes should take less time than running distributed Bellman–Ford.

12.3.4 Simulation Results

12.3.4.1 Static Networks

We use the wireless network model presented in Section 12.2.2 for our simulations, because wireless networks are a primal application for network coding. The random wireless networks are set up in a 10×10 square with a rate of connectivity $r = 3$. We run the distributed algorithm to determine the minimum energy subgraphs on

these networks for multicast connections with unit rate. Here, the energy required to transmit at unit rate to a distance d is taken to be d^2. We assume that there is no collision or interference in the network. Simulation results showed that the standard subgradient method has a better convergence time as compared to the incremental subgradient method, thus, in this section, we only present results for the standard method for Algorithm A.

Figure 12.3 shows the average convergence performance for the proposed algorithms for networks with 30/50 nodes and 4/8 terminals in the multicast. The step sizes used in the subgradient method are $\theta[n] = n^{-\alpha}$ with $\alpha = 0.8$ for $n = 0, 1, \ldots$. For the modified primal recovery method, the parameter N_a is set to 30. As we can see, the two primal cost curves coincide for the first 30 iterations, and after that, the modified method converges to the optimal value faster than the original method.

To compare the performance of the proposed scheme to the cost of multicast when network coding is not used, we use the multicast incremental power (MIP) algorithm described in [23], which is a centralized heuristic algorithm to perform minimum-energy multicast in wireless networks. For the same setting, the average cost values for the multicast given by the MIP algorithm are also shown in Fig. 12.3. As can be seen, in both cases, even the initial high-cost values from our distributed algorithms are lower than that from the centralized MIP algorithm. Moreover, in fewer than 50 iterations, the cost of the multicast using modified primal recovery is within 5% higher than the optimal value. Therefore, in a small number of iterations, the decentralized subgraph optimization algorithms yield solutions to the multicast problem with energy significantly lower than that for multicast without network coding even if a centralized scheme is used.

To have a feel of how Algorithm B would have performed in the above scenario, we observe in Fig. 12.3(a) that after 200 iterations, the cost difference between Algorithm A and the optimal value is about 0.5. For algorithm B to arrive at a neighborhood of the optimal solutions of this size, the step size should be smaller than 0.016 even if we use a very small $C = 5$. With this step size, it would take Algorithm B thousands of iterations to arrive at where Algorithm B is in 200 steps. Therefore, for all our simulations, we will use Algorithm A only.

12.3.4.2 Dynamic Networks

To illustrate the performance of our algorithms in dynamic networks, we use random networks with mobile nodes. The mobility model used in our simulations is the random direction mobility model [27], where each node selects a random direction between $[0,\ 2\pi]$ and a random speed between [minspeed, maxspeed]. A node travels to the border of the simulation area in that direction, then randomly chooses another valid direction and speed, and continues the process. Note that our algorithms are applicable to all mobility models, and we have chosen this specific one for its simplicity.

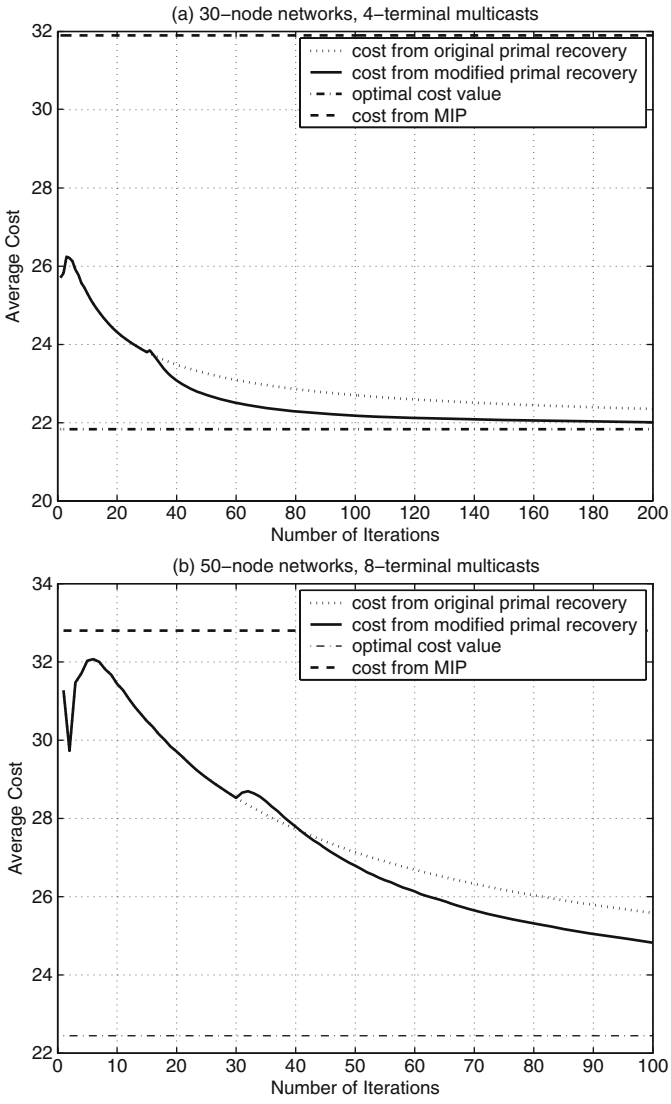

Fig. 12.3 Average cost of random 4/8-terminal multicasts in 30/50-node wireless networks, using the decentralized subgraph optimization algorithms and centralized MIP algorithm. For modified primal recovery method, $N_a = 30$

In our studies, we assume that the nodes are traveling at a speed that is slow relative to the node computation speed and link transmission rate. Under such assumptions, we consider the movement of the nodes in small discrete steps, and between each step, the set of links in the network and their costs are considered constant. We refer to the period between two discrete steps as a "static period,"

and let the number of subgraph optimization iterations performed within each static period be N_s.

We ran simulations for the various methods presented in Section 12.3.1. For dual variable initialization, we only present results based on the projection method, since it gives the best performance. Here, each node has a random speed in the interval $[0, 0.1]$ units/static period. We choose this range because the steps taken by the nodes with such speeds are relatively small compared to r, and our assumption that the network is static between steps is valid. Also, this is a relative speed of the nodes with respect to the static period, and we can vary N_s to simulate different actual speeds of the nodes.

To illustrate the typical performance of the subgraph optimization scheme in a mobile wireless network, Fig. 12.4 shows the costs for each iteration for an instance of the multicast problem. As expected, if we flush the memory of $\{\hat{x}[n]\}$ at the end of each static period, and start accumulation for the primal cost afresh, the cost of the multicast is very spiky. On the other hand, if old $\{\hat{x}[n]\}$ values are used when they are feasible, the primal cost is usually much smoother. Of course, if node movement renders the old $\{\hat{x}[n]\}$ values infeasible, we have no choice but to start afresh, and the curves for original primal recovery and look-back primal recovery coincide (as in the second static period in Fig. 12.4).

In Fig. 12.5, we show simulation results under different network and multicast settings, and for nodes with different speeds. The parameter N_a used in the modified primal recovery is set to 20. First, we compare the performance of the three options to recover primal solutions. Under the same settings, look-back primal recovery

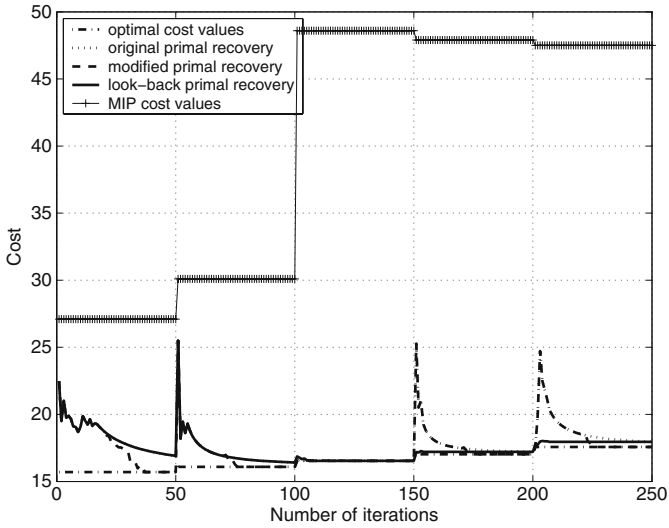

Fig. 12.4 Cost of a random four-terminal multicast in a 30-node mobile wireless network, with $N_s = 50$, under various algorithms. For the modified primal recovery method, we used $N_a = 20$ and, for the look-back primal recovery method, we used $N_a = 50$

gives the lowest average cost, followed by modified and original primal recovery. We also observe that when the same methods are used, the faster the node moves, the higher the average primal cost is, owing to the lack of time for the algorithm to converge. Also, a network with more nodes or a multicast with more terminals makes convergence of the decentralized algorithm slower, and thus results in higher average primal cost.

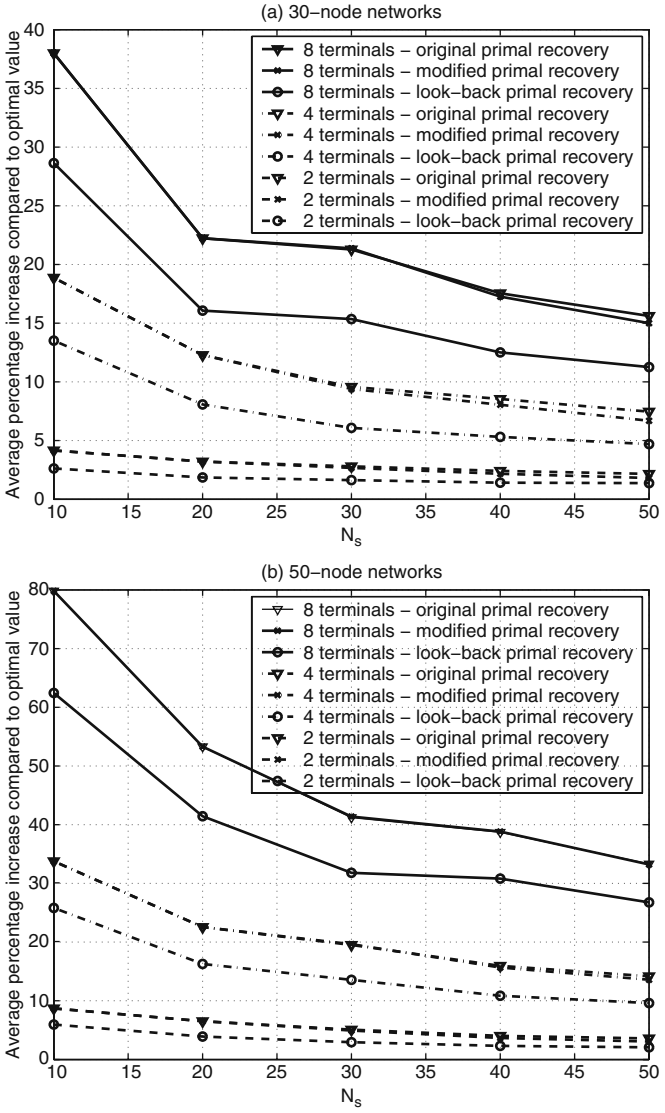

Fig. 12.5 Extra energy required for multicasts in mobile wireless networks using decentralized subgraph optimization scheme in terms of percentage of the optimal value

The simulation results have shown that the decentralized subgraph optimization scheme is robust in mobile wireless networks when the nodes are moving slowly relative to the computation and message exchange rate of the nodes. On average, it can track the changes in the optimal value closely, and, in most cases, requires lower energy for multicast than MIP even though the nodes are mobile and computation is done at each node in a distributed manner.

12.4 Min-cost Subgraph Algorithms for Dynamic Multicasts

For the dynamic multicast problem, there are two extreme cases. On one hand, we can simply find the new optimal subgraph whenever there is an update to the multicast group, and replace the existing subgraph with this new one. In this case, users in the group will experience a lot of disruptions, but the cost of the multicast is always kept minimal. On the other hand, we can enforce that no link or code rearrangement is allowed for all existing users throughout the multicast session. In this case, users enjoy uninterrupted services but, in general, the subgraph used will deviate further and further away from the optimal one. In this section, we present one algorithm to solve the nonrearrangeable version of the dynamic multicast problem, and three algorithms for the rearrangeable version. Simulation results show that one of the rearrangeable algorithms we propose, the α-scaled algorithm, can be used to strike a balance between cost and frequency of user disturbances in a distributed manner. Although we present our algorithms based on wireline networks, they can be easily extended to wireless networks.

12.4.1 Nonrearrangeable Algorithm

For simplicity, we assume that the rate of the multicast is lower than the capacity of the links, which is generally the case in current wireline networks. In a multicast session, a source node s transmits to a group of terminal nodes T, and the group changes over time. We refer to each change of the membership of the multicast group (either an addition or a removal of a terminal node) as an *online step*. The network model and problem formulation is the same as that in Section 12.2.1. This LP problem (12.1) can be solved by a number of methods, both centrally (e.g., Simplex method) and distributedly (e.g., subgradient method in Section 12.3.1). For the rest of this chapter, we denote any centralized/distributed algorithm that solves the LP problem as LP_{cent}/LP_{dist}, respectively.

For the dynamic multicast problem, the initial multicast subgraph is set up by solving (12.1). If we allow complete rearrangement, we can simply solve this problem again at each online step. However, to solve the nonrearrangeable version of the dynamic multicast problem, we have to prevent link and code rearrangements from happening. To meet the no-link rearrangement requirement, we basically need

to make sure that the existing users still use the same path(s) for the multicast when the set T changes over time. This can be achieved by setting the cost of the links in the current subgraph G_c to zero. If the capacity of a link is larger than the rate used for the multicast, then the link is split into two virtual links, one with capacity equal to the rate used for the multicast and cost zero and the other with the remaining capacity and cost unchanged. For example, if link (i, j) has capacity $c_{ij} = 2$ and rate of flow $z_{ij} = 1$ for the multicast, then nodes i and j treat link (i, j) as two parallel links with capacities 1 each, and one of them has cost of 0, and the other one has cost of a_{ij}. After doing this, the current multicast subgraph becomes "free," and doing optimization on this new cost assignment will always lead to using the same path(s) to serve the existing users in the new subgraph. Therefore, link rearrangements are avoided.

One problem with the above method is that some links not necessary for the new terminal set might be included in the new subgraph after a removal of a terminal. This is because all links in the old subgraph are free, and some of these links might still be included in the solution to the LP problem even though they are not necessary in performing the multicast to the new terminal set. To solve this problem, instead of setting their costs to 0, we can set the cost of the used links to a small value ϵ, so that no extra link would be included in the optimal solution, and, at the same time, the used links are still almost free as compared to the other links.

As for code rearrangements, we want to prevent the usage of new links that go into existing nodes of the subgraph. To do that, each node in the subgraph can scan through its incoming links, and sets the cost of those unused links to a very large value, M. Again, if the capacity of an incoming link is not fully used in the current subgraph, we can split it into two parallel virtual links as above. These nodes then send the new costs of its incoming links to their corresponding tail nodes, and the new high costs can prevent these links from being used.

After making these changes to the link costs, when an online step occurs, we can simply run LP_{dist} again with the new costs, and obtain a feasible subgraph for the new multicast group without any link or code rearrangements. This algorithm is summarized in Fig. 12.6.

```
node i

nodeUsed = 0
for all (j,i) ∈ A
   if (j,i) ∈ G_c
      a_ji = ε
      nodeUsed = 1
   end
end
if nodeUsed = 1
   for all (j,i) ∈ A
      if (j,i) ∈ G_c     a_ji = M
   end
end
call LP_dist
```

Fig. 12.6 Nonrearrangeable algorithm

The above algorithm can be complicated due to the splitting of physical links into parallel virtual links. This requires more processing at the nodes and more coordination between the end nodes of the links. In addition, in the nonrearrangeable solution of the dynamic multicast problem, it is inevitable that the subgraph used would deviate further and further from the optimal subgraph. This is because the no-rearrangement requirement forces the subgraph we use as close as possible to the initial subgraph. Thus, when the multicast group changes further and further away from the original group over time, our multicast subgraph becomes more and more suboptimal.

To simplify this algorithm and keep the cost of the multicast low, we may need to make some compromise and allow some rearrangements. In the next section, we present three such heuristic algorithms.

12.4.2 Rearrangeable Algorithms

12.4.2.1 Algorithm for Minimizing Link Rearrangement (MLR)

One way to simplify the nonrearrangeable algorithm is to focus on eliminating link rearrangement only, and ignore code rearrangement. This can be easily done by setting the used link costs to a very small value ϵ after each online step as in the nonrearrangeable algorithm, and call LP_{dist} to solve the new LP problem. This algorithm, which we call the MLR (minimal link rearrangement) algorithm, is shown in Fig. 12.7.

The motivation for this algorithm comes from the observation that the complication of splitting links into used and unused portions arises when we have a non-tree subgraph, and things would be much simpler if we only have to deal with trees. This is because, in trees, each node only has one incoming link, and it has full information of the multicast. Thus, there is no worry about code rearrangement.

Notice that once the multicast subgraph becomes a tree, it will remain as a tree through the rest of the online steps. To see this, consider addition of a new node to the multicast group. Since the original subgraph G_c is considered "free" and each node in G_c has full information of the multicast, the new terminal node only needs to find the shortest path from any node in G_c to itself, and attach itself to the subgraph. As for the removal of a terminal, only a part of the tree may be removed, and the remaining graph should still be a tree. Since at every step, if G_c is not a tree, there is

```
node i

for all (j,i) ∈ A
   if (j,i) ∈ G_c
      a_ji = ε
   end
end
call LP_dist
```

Fig. 12.7 MLR algorithm

some positive probability that it will become a tree, and once it evolves into a tree, it will stay that way till the end of the multicast. Therefore, if we keep running the dynamic multicast session, the probability that we are dealing with trees goes to 1.

In addition, simulations on practical networks show that in more than 98% of the time, we do get the optimum Steiner tree at start-up. Therefore, we can focus on link rearrangements only and use the MRL algorithm. This algorithm still works if the initial subgraph is not a tree, the only difference is that we cannot guarantee that there will not be any rearrangements in such cases.

12.4.2.2 Algorithm for Limiting Multicast Cost (LMC)

If we use the MLR algorithm, it is expected that as time goes on, the subgraph used for multicast will move further and further away from the actual optimal subgraph for the current set of terminal nodes. As an alternative, we might want to introduce occasional rearrangements in order to keep the cost of the multicast close to optimal. We introduce the LMC algorithm, shown in Fig. 12.8, to do this. In this algorithm, the nodes run two programs in parallel, one of which generates the subgraph with no rearrangement using the algorithm presented above. We call this subgraph the *no-change subgraph* G_{nc}, and the cost of this subgraph C_{nc}. The other program keeps track of the optimal subgraph, G_{opt}, for the current set of multicast terminals, and the cost of G_{opt} is C_{opt}. At each step, the cost of the two subgraphs are compared, and if the cost of the no-change subgraph is higher than the optimal graph by a certain factor, β, we switch to the optimal subgraph. Using this method, we can control the trade-off between the frequency of disturbances to the users and the cost of the subgraph used for the multicast by changing the value of β. However, this method requires the nodes to keep track of two subgraphs, and centralized coordination is needed to compare the costs and make the nodes switch between two subgraphs simultaneously.

12.4.2.3 α-Scaled Algorithm

We now present a simple approximate algorithm that can trigger "auto-switching" between G_{nc} and G_{opt} in a distributed manner. Instead of assigning a very small cost to the used links as in the MLR algorithm, we can use a scaled value of the

```
call LP_cent to compute C_opt and G_opt
for all (j,i) ∈ A
    if (j,i) ∈ G_c
        a_ji = ε
    end
end
call LP_cent to compute C_nc and G_nc
if C_nc > C_opt × (1+β)
    use G_opt for the multicast
else
    use G_nc for the multicast
end
```

Fig. 12.8 LMC algorithm

original cost, i.e., for an existing link (i, j) in the subgraph, we use αa_{ij} as its cost in the future computations as long as it stays in the subgraph, where α is a scaling factor between 0 and 1. If $\alpha = 0$, it is the same as the MLR algorithm; and if $\alpha = 1$, we will be using the optimal subgraph every time. We refer to this algorithm as the *α-scaled algorithm*, and it is shown in Fig. 12.9.

To see why this heuristic works and how the constants α and β are related, consider the case of removal of a terminal node. The LMC algorithm compares the values of C_{nc} and $(1+\beta)C_{opt}$ and picks the lower of the two. Since G_{opt} may overlap with the existing subgraph from before the online step, we assume the cost of this overlapping part of the subgraph is C_{ol} and the cost of the rest of the optimal subgraph is C_{others}. Thus, the comparison is equivalent to

$$\frac{1}{1+\beta} \times C_{nc} \gtrless C_{ol} + C_{others}. \tag{12.25}$$

On the other hand, in the α-scaled algorithm, we are effectively choosing the lower cost between these two:

$$\alpha \times C_{nc} \gtrless \alpha \times C_{ol} + C_{others}. \tag{12.26}$$

If we set α to $1/(1+\beta)$, we can see that (12.25) and (12.26) are very similar except the first term on the right-hand side. By scaling the existing link costs by α, we can satisfy the requirement that the cost of the subgraph used never goes over $(1+\beta)C_{opt}$, but owing to the scaling factor α on C_{ol}, the approximate algorithm switches to the optimal subgraph more often than required by β. Using similar analysis, we have the same results for the case of addition of a terminal.

Thus, using an appropriate α to scale the costs of the used links, the optimization can trigger auto-switching between the two subgraphs, thus keeping the cost of the multicast low. In addition, we can make α a time-varying variable. In general, when a link is first added into the subgraph, it is likely that it will remain there for a while. However, the probability that the link remains in the optimal subgraph decreases with the online steps. To capture this characteristic, we can use a lower value for α for the first few online steps after a new link is added, and increase α gradually later on. Also, in a practical network, it may not be desirable to make back-to-back changes to the link connections, i.e., addition of a link to the multicast subgraph followed by an immediate removal of it in the next step. We can reduce the occurrence of such events by setting the α of new links to 0 for a few steps before raising it to the normal value of $1/(1+\beta)$.

```
node i

for all (j,i) ∈ A
    if (j,i) ∈ G_c
        a_ji = α × a_ji
    end
end
call LP_dist
```

Fig. 12.9 α-Scaled algorithm

12.4.3 Simulation Results

We first present simulation results for the MLR algorithm. The network topologies used in the simulations are obtained from the Rocketfuel project [28]. In each simulation, we start with a multicast from a random source to a set of 10 random terminals. Subsequently, in each online step, we first randomly decide whether there is an addition or removal of terminal, and then randomly select a terminal to add/remove based on that decision. Figures 12.10 and 12.11 show the average increase of cost of the no-change subgraph compared to the cost of the optimal subgraph in terms of percentage of C_{opt}. The network topology used for Figs. 12.10 and 12.11 are backbones for Exodus (United States) and EBONE (Europe), respectively. As expected, the extra cost of the no-change subgraph grows approximately linearly with the online steps. In addition to the average curve, we also show the data points for each instance of the simulation in both Figs. 12.10 and 12.11. Note that there are cases when the cost of the no-change subgraph is as much as 60% higher than the optimal cost after 20 steps.

This undesirable phenomenon motivates the usage of the α-scaled algorithm. Figure 12.12 shows the simulation results for using the α-scaled algorithm on the network used in Fig. 12.10. Here, we aim to control the cost of the subgraph used to within $\beta = 30\%$ away from that of the optimal subgraph; thus, we use $\alpha = 0.75$. The average curve in Fig. 12.10 for the MLR algorithm is also shown here for comparison. The α-scaled algorithm provides lower cost for the multicasts as compared to the MLR algorithm. More importantly, the cost difference between the

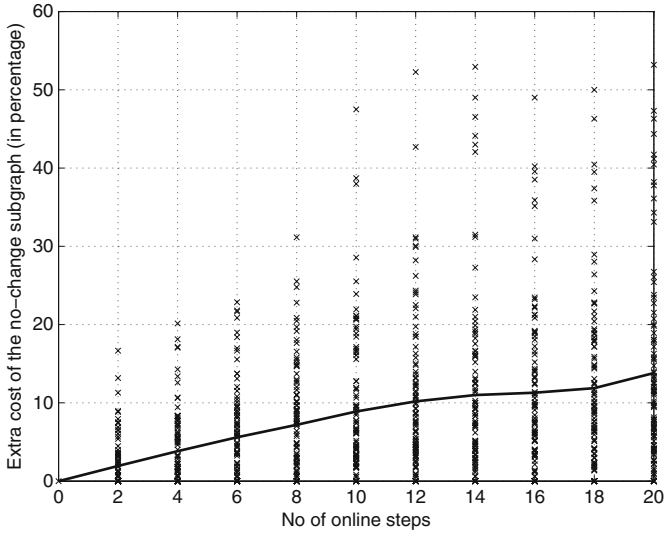

Fig. 12.10 Extra cost of the multicast subgraph generated by the MLR algorithm in terms of percentage of C_{opt} for the Exodus network. We are showing both the individual data points for each trial and the average curve

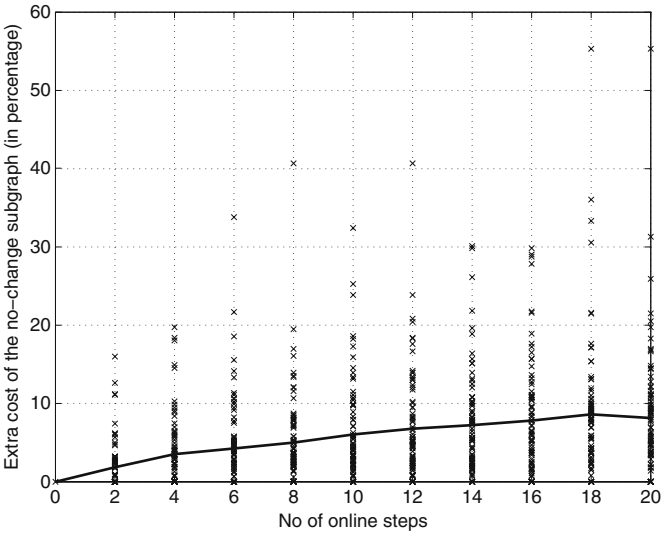

Fig. 12.11 Extra cost of the multicast subgraph generated by the MLR algorithm for the EBONE network in terms of percentage of C_{opt}

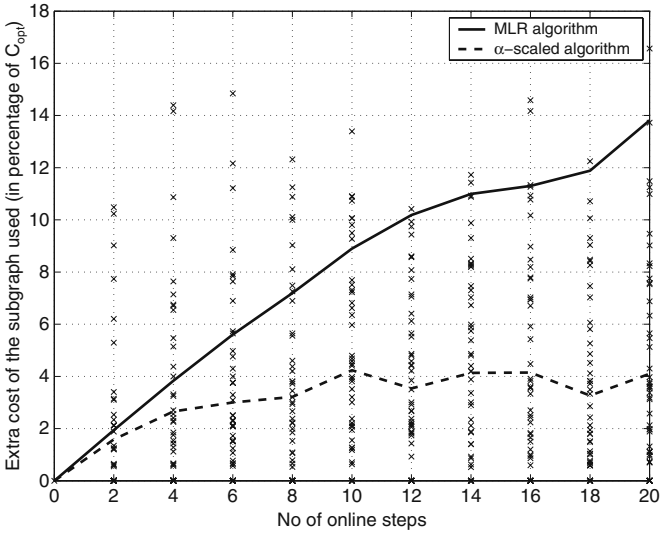

Fig. 12.12 Extra cost of the multicast subgraph generated by the α-scaled algorithm with $\alpha = 0.75$ and the MLR algorithm in terms of percentage of C_{opt}, on the Exodus network. We are also showing the individual data points for each trial of the α-scaled algorithm

subgraph used and C_{opt} for the α-scaled algorithm is roughly constant after a while, and it does not grow over time. Of course, there is a price for this gain, which is the occasional switching from the no-change subgraph to the optimal subgraph. In this

case, the average switching probability is 11.7%, which means, out of a hundred online steps, there are about 12 times when the existing users might experience disturbances to their transmissions. Furthermore, if we look closely at the data points for individual instances, we can see that, actually, none of the instance has gone over 20% higher than the optimal one. This is consistent with our discussion in Section 12.3 about the values of α and β. Therefore, if we want to have $\beta = 30\%$, we can use a lower value for α.

Finally, Fig. 12.13 shows the simulation results for the same network setup with different α values. As we can see, the higher the α value, the lower the average cost of the subgraph. At the same time, higher α values lead to higher switching rate. We observed that when α is equal to 0.5, the cost of the subgraph used is kept at around 9% higher than the optimal cost, whereas the switching probability is only 2.05%. Therefore, by selecting the α-value properly, we can keep the cost of the multicast close to optimal during the multicast session while causing few disturbances to the existing users.

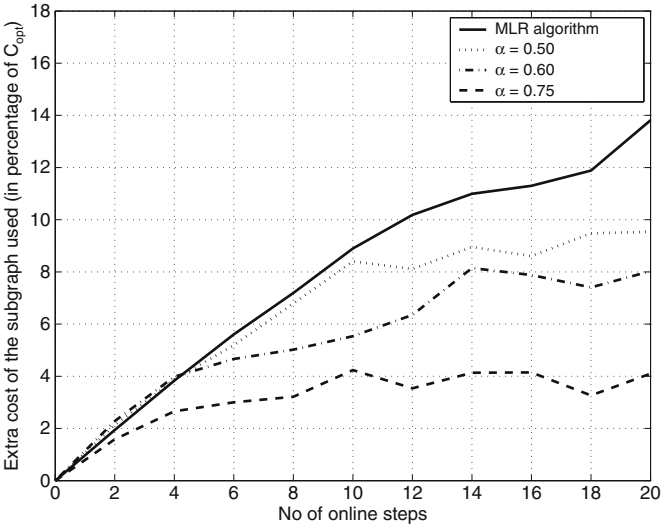

Fig. 12.13 Extra cost of the multicast subgraph generated by the α-scaled algorithm with various α-values, on the Exodus network

12.5 Conclusions

Subgraph optimization is an important problem in performing multicast with network coding. In this chapter, we studied the algorithms that solve this optimization problem for both static and dynamic multicasts. For static multicast, we presented distributed subgradient algorithms to find the min-cost subgraph and examined their

convergence rate. Using the special structure of the network coding problem, we showed that with appropriately chosen step sizes, the dual problem converges to a neighborhood of the optimal solution linearly. Since the physical flow variables are decoupled from the dual iterations, we can obtain a feasible primal solution in each iteration. The convergence rate of the primal solutions is $O(1/n)$. We also proposed various heuristics for dual variable initialization and primal solution recovery to further improve the convergence performance. Simulation results show that the subgradient method produces significant reductions in multicast energy compared to centralized routing algorithms after just a few iterations. Moreover, the algorithm is robust to changes in the network and can converge to new optimal solutions quickly as long as the rate of change in the network is slow compared to the speed of computation and transmission.

For dynamic multicasts, in order to characterize the disturbances to users caused by the changes in the multicast subgraph, we introduced the concepts of link rearrangement and code rearrangement. We proposed both nonrearrangeable and rearrangeable algorithms for the dynamic multicast problem, and used simulation results to show that the α-scaled algorithm can effectively bound the growth of the multicast cost without causing too many disturbances to existing users.

Acknowledgments This work was supported by the BAE Systems National Security Solutions Inc. under grant "Control Based Mobile Ad-Hoc Networking Program (CBMANET)" (060786).

References

1. R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.
2. S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371–381, February 2003.
3. R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, October 2003.
4. S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. G. M. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, vol. 5, no. 6, pp. 1973–1982, June 2005.
5. T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, October 2006.
6. D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. K. T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2608–2623, June 2006.
7. T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. of the 41th Annual Allerton Conference on Communication, Control, and Computing*, October 2003.
8. T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. 2003 IEEE International Symposium on Information Theory (ISIT'03)*, Yokohama, Japan, June–July 2003.
9. P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. of the 41th Annual Allerton Conference on Communication, Control, and Computing*, October 2003.

10. T. Ho, B. Leong, M. Médard, R. Koetter, Y.-H. Chang, and M. Effros, "On the utility of network coding in dynamic environments," in *Proc. 2004 International Workshop on Wireless Ad-hoc Networks (IWWAN'04)*, 2004.
11. Y. Wu, P. A. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1906–1918, November 2005.
12. D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving minimum cost multicast: A decentralized approach based on network coding," in *Proc. IEEE Infocom*, vol. 3, March 2005, pp. 1607–1617.
13. K. Bharath-Kumar and J. M. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Trans. Commun.*, vol. 31, no. 3, pp. 343–351, March 1983.
14. B. M. Waxman, "Routing of multicast connections," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, December 1988.
15. M. Chiang, "Nonconvex optimization of communication systems," in *Advances in Mechanics and Mathematics, Special Volume on Strang's 70th Birthday*, D. Gao and H. Sherali, Eds. Springer, New York, NY, U.S.A., 2008.
16. H. D. Sherali and G. Choi, "Recovery of primal solutions when using subgradient optimization methods to solve lagrangian duals of linear programs," *Oper. Res. Lett.*, vol. 19, pp. 105–113, 1996.
17. T. Larsson, M. Patriksson, and A. Strömberg, "Ergodic primal convergence in dual subgradient schemes for convex programming," *Math. Program.*, vol. 86, pp. 283–312, 1999.
18. K. C.'Kiwiel, T. Larsson, and P. O. Lindberg, "Lagrangian relaxation via ballstep subgradient methods," *Math. Oper. Res.*, vol. 32, no. 3, pp. 669–686, August 2007.
19. A. Nedić and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," MIT LIDS, Tech. Rep., 2007.
20. D. S. Lun, M. Médard, and D. R. Karger, "On the dynamic multicast problem for coded networks," in *Proc. of WINMEE, RAWNET and NETCOD 2005 Workshops*, April 2005.
21. M. Imase and B. M. Waxman, "Dynamic steiner tree problem," *SIAM J. Discrete Math.*, vol. 4, no. 3, pp. 369–384, August 1991.
22. S. Raghavan, G. Manimaran, and C. S. R. Murthy, "A rearrangeable algorithm for the construction delay-constrained dynamic multicast trees," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 514–529, August 1999.
23. J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "Energy-efficient broadcast and multicast trees in wireless networks," *Mobile Netw. Appl.*, vol. 7, pp. 481–492, 2002.
24. D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, Nashua, NH, U.S.A, 1995.
25. A. Nedić, "Subgradient methods for convex minimization," Ph.D. dissertation, Massachusetts Institute of Technology, June 2002.
26. J. V. Burke and M. C. Ferris, "Weak sharp minima in mathematical programming," *SIAM J. Control Optim.*, vol. 31, no. 5, pp. 1340–1359, September 1993.
27. T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Commun. Mob. Comput.*, vol. 2, no. 5, pp. 483–502, August 2002.
28. The rocketfuel project. [Online]. Available: www.cs.washington.edu/research/networking/rocketfuel