

Chapter 2

Distributed Optimization in Networking: Recent Advances in Combinatorial and Robust Formulations

Minghua Chen and Mung Chiang

2.1 Introduction

Optimization has become an essential modeling language and design method for communication networks. It has been widely applied to many key problems, including power control, coding, scheduling, routing, congestion control, content distribution, and pricing. It has also provided a fresh angle to view the interactions across a network protocol stack as the solutions to an underlying optimization problem. A unique requirement for optimization in networks is that the solution algorithm must be distributed. This has in turn motivated the development of new tools in distributed optimization. Many of these results have been well documented. In this chapter, we turn to a sample of three recent results on some of the challenging new issues, centered around the need to tackle combinatorial or robust optimization formulation through distributed algorithms.

2.1.1 *Garg–Konemann Framework for Fractional Packing Linear Programming Problems*

A number of system design problems can be formulated as fraction packing linear programming problems. Such problems often come with exponential number of variables, and are challenging to solve. The example we will focus on in Sect. 2.2

M. Chen (✉)

Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong
e-mail: minghua@ie.cuhk.edu.hk

M. Chiang

Department of Electrical Engineering, Princeton University, Princeton, NJ, USA
e-mail: chiangm@princeton.edu

is peer-to-peer (P2P) streaming capacity problem [1–3]: given a P2P network with a source node and a set of receivers, how to embed a set of multicast trees spanning the receivers and to determine the amount of flow in each tree, such that the sum of flows over these trees is maximized?

Garg and Konemann [4] presented a framework for solving these challenging problems approximately in polynomial time. The intuitive observation behind the framework is similar to those of column generation [5–7]: although solving the problem exactly may require exploring the space spanned by all the exponential number of variables, solving the problem approximately only requires tuning a polynomial-size subset of them. The column generation technique does not specify how to find one such subset of the variables, as they are usually problem-specific. Interestingly, for all fractional packing problems, the Garg–Konemann framework provides a systematic way to find one such subset of the variables.

2.1.2 Markov Approximation for Combinatorial Optimization

Many important network resource allocation problems are combinatorial in nature. The objective of these problems is to maximize a system-wide performance metric, which involves enumerating all possible configurations of many independent entities. Problems of this kind appear in various domains, including wireless networking [8–10], P2P networking [2, 3], content distribution [11], and cloud computing [12, 13].

We observe a gap between the known and the desired solutions for many of these combinatorial problems. On one hand, exact solutions are computationally prohibitive. Research thus focuses on finding efficient approximation algorithms. However, most of these algorithms only allow *centralized* implementations.

On the other hand, system designers usually prefer distributed algorithms. Distributed algorithms are more adaptable to network resource fluctuation and users joining and leaving as compared to centralized ones.

In Sect. 2.3, we present a general Markov approximation technique that allows us to approximately solve combinatorial optimization problems in a distributed manner. This also addresses the computational complexity issue to a certain extent because the distributed implementation often allows parallel processing by different elements. To demonstrate the usefulness of the technique, we apply it to design a distributed streaming algorithm for P2P systems that achieves close-to-optimal performance.

2.1.3 Distributed Robust Optimization

Robustness of optimization models for network problems in communication networks has been an under-explored topic. Most existing algorithms for solving

robust optimization problems are centralized, thus not suitable for networking problems that demand distributed solutions. In Sect. 2.4, we summarize a first step toward a systematic theory for designing distributed *and* robust optimization models and algorithms. We first discuss several models for describing parameter uncertainty sets that can lead to decomposable problem structures and thus distributed solutions. These models include ellipsoid, polyhedron, and D -norm uncertainty sets. We then apply these models in solving a robust rate control problem in wireline networks.

2.2 P2P Streaming Capacity

2.2.1 Problem Settings

Consider a P2P network as a P2P graph $G = (V, E)$, where each node $v \in V$ may be the source, a receiver, or a helper that serves only as a relay; each edge $e = (u, v) \in E$ represents a neighboring relationship between vertices u and v . We assume that data rate bottlenecks only appear at node uplinks. Denoting by $C(v)$ the uplink capacity of node v , we have for each node v ,

$$\sum_{u \in V} x_{vu} \leq C(v), \quad (2.1)$$

where x_{vu} is the rate node v transmits to node u . This assumption is widely adopted in the P2P literature because in today's Internet, access links are the bottlenecks rather than backbone links, and uplink capacity is several times smaller than downlink capacity in access networks.

We consider the single session scenario in this chapter.¹ In this scenario, the session content originates from one source s and is distributed to a given set of receivers R , possibly using a set of helpers H . A packet in the session stream starts from the source s , and traverses over all nodes in R , and some nodes in H —the traversed paths form a Steiner tree in the overlay graph G . Here a Steiner tree refers to a tree that connects the sender and all the receivers and allows the sender to reach any receiver over it. Different packets in the same session may traverse different trees, and we call each tree a *sub-tree*, and call their superposition a *multi-tree*.

There are P2P protocol constraints on sub-trees. The most frequently encountered one is degree constraint. For example, in the widely used P2P protocol of BitTorrent, although one node has 30–50 neighbors in G , it can upload to at most five of them as peers. This gives an outgoing degree bound for each node and constrains the construction of the trees. Degree bounds always apply to receivers and helpers and,

¹As compared to single-session scenario, the multi-session scenario involves multiple sessions competing for the underlying physical link capacities, and one has to take the fairness consideration into account when formulating the problem. We refer interested readers on the multi-session study to [1].

in some scenarios, to the source as well. Let $m_{v,t}$ be the number of outgoing edges of node v in tree t , and the bound be $M(v)$: $m_{v,t} \leq M(v), \forall t$. We denote by T the set of all allowed sub-trees: trees that satisfy the constraints such as the degree bounds.

For each tree $t \in T$, we denote by y_t the rate of the sub-stream supported by this tree. We say the session has a rate $r = \sum_{t \in T} y_t$ if all the receivers obtain the streaming contents at a rate of r or above. A rate is called *achievable* if there is a multi-tree in which all trees satisfy the topology constraint ($t \in T$) and transmission rates satisfy the uplink capacity constraint. We define *P2P streaming capacity* as the largest achievable rate.

2.2.2 Formulation

The P2P streaming capacity problem for single session is formulated as follows.

2.2.2.1 Single-Session (Primal) Problem

$$\text{maximize} \quad r = \sum_{t \in T} y_t \quad (2.2)$$

$$\text{subject to} \quad \sum_{t \in T} m_{v,t} y_t \leq C(v), \forall v \in V \quad (2.3)$$

$$y_t \geq 0, \forall t \in T \quad (2.4)$$

$$\text{variables} \quad y_t, \forall t \in T \quad (2.5)$$

For those trees not selected in the solution, their rates y_t are simply zero.

The dual problem associates a nonnegative variable $p(v)$, interpreted as price, with each node v corresponding to constraint (2.3). It can be derived to be the following problem.

2.2.2.2 Single-Session Dual Problem

$$\text{minimize} \quad \sum_{v \in V} C(v) p(v) \quad (2.6)$$

$$\text{subject to} \quad \sum_{v \in V} m_{v,t} p(v) \geq 1, \forall t \in T, \quad (2.7)$$

$$p(v) \geq 0, \forall v \in V \quad (2.8)$$

$$\text{variables} \quad p(v), \forall v \in V \quad (2.9)$$

We can interpret the dual problem this way: $p(v)$ is the per unit flow price for any edge outgoing from v . If node v uploads with full capacity, the incurred cost is $p(v)C(v)$. There are $m_{v,t}$ connections outgoing from node v in tree t , and thus the

total tree price for tree t , which is defined as the sum of prices in any edge in tree t , is $\sum_{v \in V} m_{v,t} p(v)$. Therefore, the dual problem is to minimize the total full capacity tree cost given that the tree price is at least 1, and the minimization is over all possible \mathbf{p} , where $\mathbf{p} := \{p(v), \forall v \in V\}$ is the price vector. For notational simplicity, we use $p(\cdot) : V \rightarrow R^+ \cup \{0\}$ to represent \mathbf{p} .

In general, the number of trees we need to search when computing the right multi-tree grows exponentially in the size of the network. This dimensionality increase is the consequence of turning a difficult graph-theoretic, discrete problem into a continuous optimization problem. Hence, the primal problem can have possibly exponential number of variables and its dual can have an exponential number of constraints, neither of which is suitable for direct solution in polynomial time. However, as detailed in the next subsection, the above representations turn out to be very useful to allow a primal–dual update outer loop that converts the combinatorial problem of multi-tree construction into a usually simpler problem of smallest price tree (SPT) construction.

2.2.3 Algorithm and Performance

Adapting the technique for solving the maximum multi-commodity flow problem in [4], we design an iterative combinatorial algorithm that solves the primal and dual problems approximately, where tree-flows are augmented in the primal solution and dual variables are updated iteratively. Our algorithm constructs peering multi-trees that achieve an objective function value within $(1 + \zeta)$ -factor of optimal.

For a given tree t and prices $p(\cdot)$, let $Q(t, p)$ denote the left-hand side (LHS) of constraint (2.7), which we call the *price* of tree t . A set of prices $p(\cdot)$ is a feasible solution for the dual program if and only if

$$\min_{t \in T} Q(t, p) \geq 1.$$

The algorithm works as follows. Start with initial weights $p(v) = \frac{\delta}{C(v)}$ for all $v \in V$. Parameter δ depends on ζ and is described in more detail later. Repeat the following steps until the dual objective function value becomes greater than 1:

1. Compute a tree \bar{t} for which $Q(t, p)$ is minimum. We call \bar{t} a *SPT* problem, algorithms for which are developed in next section.
2. Send the maximum flow on this tree \bar{t} such that uplink capacity of at least one internal node is saturated. Let $I(\bar{t})$ be the set of internal nodes in tree \bar{t} . The flow sent on this tree is

$$y = \min_{v \in I(\bar{t})} \frac{C(v)}{m_{v,\bar{t}}}. \quad (2.10)$$

Fig. 2.1 The primal–dual algorithm for single-session P2P streaming capacity computation

<p>Primal-Dual Algorithm: Single-Session $p(v) \leftarrow \frac{\delta}{C(v)}, \text{flow}(v) \leftarrow 0, \forall v \in V, Y \leftarrow 0, D \leftarrow 0$</p> <p>while $D < 1$ Pick tree $t \in T$ with the smallest $Q(t, p)$ $y \leftarrow \min_{v \in I(t)} C(v)/m_{v,t}$ $\bar{t} \leftarrow \arg \min_{v \in I(t)} C(v)/m_{v,t}$ $\text{flow}(v) \leftarrow \text{flow}(v) + ym_{v,\bar{t}}, \forall v \in I(\bar{t})$ $Y \leftarrow Y + y$ $p(v) \leftarrow p(v)(1 + \epsilon \frac{m_{v,\bar{t}}y}{C(v)})$ $D \leftarrow \sum_{v \in V} C(v)p(v)$ end while</p> <p>Compute scaling factor $\alpha \leftarrow \max_{v \in V} \frac{\text{flow}(v)}{C(v)}$; Output capacity $r^* \leftarrow Y/\alpha$;</p>
--

3. Update the prices $p(v)$ as

$$p(v) \leftarrow p(v) \left(1 + \frac{\epsilon m_{v,\bar{t}}y}{C(v)} \right), \forall v \in I(\bar{t}).$$

where ϵ depends on θ and is explained in more detail later.

4. Increment the flow Y sent so far by y .

The optimality gap can be estimated by computing the ratio of the primal and dual objective function values in each step of the above iteration, which can be terminated after the desired proximity to optimality is achieved. When the above iteration terminates, primal capacity constraints on each uplink may be violated, since we were working with the original (and not residual) uplink capacities at each stage. To remedy this, we scale down the flows uniformly so that uplink capacity constraints are satisfied.

The pseudo-code for the above procedure is provided in Fig. 2.1. Array $\text{flow}(v)$ keeps track of the traffic on uplink of node v as the algorithm progresses. The dual objective function value is tracked by variable D which is initialized to 0. After the “while” loop terminates, the maximum factor by which the uplink capacity constraint is violated on any uplink is computed as α , which divides the total flow Y , and the resulting value is output as r^* .

The theorem below states accuracy and complexity properties of the algorithm. Its proof can be found in [1].

Theorem 2.1. *For any given $\zeta > 0$, the Single-Session Primal–Dual Algorithm computes a solution with objective function value within $(1 + \zeta)$ -factor of the optimum, for algorithmic parameters $\epsilon(\zeta) = 1 - \frac{1}{\sqrt{1+\zeta}}$ and $\delta(\zeta) = \frac{1+\epsilon}{[(1+\epsilon)|V|]^{1/\epsilon}}$.*

It runs in time polynomial in the input size and $\frac{1}{\epsilon}$: $O\left(\frac{|V|\log|V|}{\epsilon^2}T_{spt}\right)$, where T_{spt} is the time to compute a SPT.

This unifying primal–dual framework works for *all* of the single session problems. The core issue now lies with the inner loop of SPT computation: can this be accomplished in polynomial time for a given price vector? This graph-theoretic problem is generally much more tractable than the original problem of searching for the multi-tree that maximizes the achievable rate. However, when the given graph G is not a full mesh, or when there are degree bounds on nodes in each tree, or when there are helper nodes, computing an SPT becomes difficult.

2.2.4 Computing SPT

For iterative algorithms, efficiently and accurately computing an SPT for a given set of node prices is the key module in each outer loop. We have developed SPT algorithms for general problem formulations over P2P graph in [1]. We present the case of general P2P graph without degree bound and with helpers in the following; we leverage special structures of the P2P graph G in the development.

We now study the cases where the given graph G is not complete and without degree bounds in the trees. We consider the case with the presence of helpers. In this case, the SPT computation problem is a minimum cost directed Steiner tree problem *with symmetric connectivity and a special structure on the costs*—the costs of all edges going out of a node are equal. We leverage these two special features to accelerate the algorithm through a graph transformation.

Let $[v_1, v_2]$ represent the pair of links connecting two neighboring nodes v_1 and v_2 . We want to find the minimum price directed Steiner tree connecting source s and a set of receivers in R . We transform the directed graph G to an undirected graph $G' = (V', E')$, which represents the adjacency between link pairs and the source node s :

- For source node s , we copy it into G' .
- For every two neighboring nodes $v_1, v_2 \in V$, we map the link pair $[v_1, v_2]$ to a node $n_{[v_1, v_2]}$ in G' .
- For every node $v_1 \in V$ in G , we map it to a series of *undirected* links connecting nodes $n_{[v_1, v_2]}$ and $n_{[v_1, v_3]}$ in G' where v_2 and v_3 are any neighbors of v_1 in G ; we set the prices of these undirected links to $p(v)$.
- In graph G' , we connect any two of nodes s and $n_{[s, v]}$, with a series of *undirected* links, where v is any neighbor of s in G ; we set the prices of these links to be $p(s)$.

An example of such transformation is illustrated in Fig. 2.2.

For the undirected graph G' , we consider the following group Steiner tree problem. For every receiver $r \in R$ in G , we group all the nodes $n_{[r, v]} \in V'$, $v \in V$, into a set, denoted by g_r . Set g_r in G' corresponds to the set of link pairs in G connecting

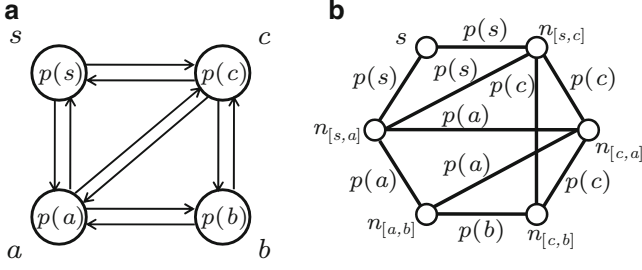


Fig. 2.2 (a) An overlay graph with source node s , receiver nodes a and b , and Steiner node c with node prices being p_s , p_a , p_b , and p_c respectively; (b) the undirected graph mapped from the overlay graph in (a); for example, the link pair between a and b in (a) maps to the node $n_{[a,b]}$ in (b), and node a maps to three links with link cost p_a in (b)

r to all its neighbors. We also construct a set g_s that contains only the source s in G' . The group Steiner tree problem in G' is to find the minimum price Steiner tree that connects at least one node from each of sets g_s and g_r , $r \in R$. Solving the group Steiner tree problem in undirected graph is NP-hard [14]. The authors in [14] propose a polynomial time algorithm that achieves an approximation factor of $\frac{1}{O(\ln N_g \ln^3 N_m)}$, where N_g is the number of groups and N_m is total number of nodes.

The following theorem, proved in [1], states that finding the minimum cost Steiner tree in G is equivalent to searching the minimum cost group Steiner tree in G' .

Theorem 2.2. *Consider finding a Steiner tree in G that connects s and all nodes in R , and searching a group Steiner tree in G' that connects at least one node from each of node sets g_s and g_r , $r \in R$, the following is true:*

1. *A directed Steiner tree in G can be mapped to a group Steiner tree in G' with the same price in polynomial time.*
2. *A group Steiner tree in G' can be mapped to a directed Steiner tree in G with the same or less price in polynomial time.*

Consequently, the optimal group Steiner tree in G' can be mapped to the optimal directed Steiner tree in G in polynomial time and vice versa. Furthermore, their prices are equal.

The minimum cost directed Steiner tree problem is hard to approximate to a factor better than $\frac{1}{\ln |R|}$ [15]. An $\frac{1}{O(\ln |R|)}$ -factor approximation algorithm that runs in polynomial time for any fixed $\varepsilon > 0$ is given in [16]. Theorem 2.2 states that the directed Steiner tree problem in graph G can be approached by studying a group Steiner tree problem in G' . We first apply the randomized algorithm proposed in [14] to G' , and get a group Steiner tree with an approximation factor of $\frac{1}{O(\ln |R| \ln^3 N_m)}$. N_m corresponds to total number of link pairs in G , and is at most $|V|^2$. We then map

this group Steiner tree to a directed Steiner tree in G . Since this mapping keeps or reduces the price, at the end we compute a directed Steiner tree in G with an approximation factor of $\frac{1}{O(\ln|R|\ln^3|V|)}$ in polynomial time. In the case where $|V| = O(|R|)$, we can compute a directed Steiner tree with an approximation factor of $\frac{1}{O(\ln^4|R|)}$ in polynomial time.

2.3 Markov Approximation for Combinatorial Optimization

2.3.1 General Framework

2.3.1.1 Problem Formulation

Consider a system with a set of users R , and a set of configurations \mathcal{F} . A network configuration $f \in \mathcal{F}$ consists of individual components using one of its local configurations. The system obtains a certain performance when it operates under configuration f , denoted by W_f . The problem of maximizing the system performance by choosing the best configuration can then be cast as the following combinatorial optimization problem

$$\mathbf{MWC} : \max_{f \in \mathcal{F}} W_f. \quad (2.11)$$

One well-known example is the neighbor selection problem in P2P streaming systems. The name of the game is to assign the “best” set of neighbors for every peer/cache, in order to form the best topology to support a streaming video at the highest possible quality [2, 3]. In the example, a local configuration of a peer is the set of neighbors it connects to, and a network configuration is a vector consisting of local configurations of all the peers. The system performance metric in the example is the video streaming rate.

For the problem **MWC**, an equivalent formulation is

$$\begin{aligned} \mathbf{MWC-EQ} : \max_{p \geq 0} \sum_{f \in \mathcal{F}} p_f W_f \\ \text{s.t.} \quad \sum_{f \in \mathcal{F}} p_f = 1, \end{aligned} \quad (2.12)$$

where p_f is the percentage of time the configuration f is in use. Treating W_f in (2.11) as the “weight” of f , the problem **MWC** is to find a maximum weighted configuration.

2.3.1.2 Log–Sum–Exp Approximation

To gain insights on the structure of the problem **MWC**, we approximate the max function in (2.11) by a differentiable function as follows:

$$\max_{f \in \mathcal{F}} W_f \approx \frac{1}{\beta} \log \left(\sum_{f \in \mathcal{F}} \exp(\beta W_f) \right) \triangleq g_\beta(W), \quad (2.13)$$

where β is a positive constant and $W \triangleq [W_f, f \in \mathcal{F}]$. This approximation is known as the convex log–sum–exp approximation to the max function. The following bound on approximation accuracy is well known.

Theorem 2.3. *For a positive constant β and n nonnegative real variables y_1, y_2, \dots, y_n , we have*

$$\begin{aligned} \max(y_1, \dots, y_n) &\leq \frac{1}{\beta} \log(\exp(\beta y_1) + \dots + \exp(\beta y_n)) \\ &\leq \max(y_1, \dots, y_n) + \frac{1}{\beta} \log n. \end{aligned} \quad (2.14)$$

Hence, $\max(y_1, \dots, y_n) = \lim_{\beta \rightarrow \infty} \frac{1}{\beta} \log(\exp(\beta y_1) + \dots + \exp(\beta y_n))$.

We summarize some important, well-known observations of $g_\beta(x)$ in the following theorem. Some of these observations were also found relevant in the context of Geometric Programming [17].

Theorem 2.4. *For the log–sum–exp function $g_\beta(W)$, we have*

- *Its conjugate function is given by*

$$g_\beta^*(\mathbf{p}) = \begin{cases} \frac{1}{\beta} \sum_{f \in \mathcal{F}} p_f \log p_f & \text{if } \mathbf{p} \geq 0 \text{ and } \mathbf{1}^T \mathbf{p} = 1 \\ \infty & \text{otherwise.} \end{cases} \quad (2.15)$$

- *It is a convex and closed function; hence, the conjugate of its conjugate $g_\beta^*(\mathbf{p})$ is itself, i.e., $g_\beta(W) = g_\beta^{**}(W)$. Specifically,*

$$\begin{aligned} g_\beta(W) &= \max_{\mathbf{p} \geq 0} \sum_{f \in \mathcal{F}} p_f W_f - \frac{1}{\beta} \sum_{f \in \mathcal{F}} p_f \log p_f \\ \text{s.t. } &\sum_{f \in \mathcal{F}} p_f = 1. \end{aligned} \quad (2.16)$$

Several observations can be made. First, by the log–sum–exp approximation in (2.13), we are implicitly solving an approximated version of the problem **MWC–EQ**, off by an *entropy* term $-\frac{1}{\beta} \sum_{f \in \mathcal{F}} p_f \log p_f$. The optimality gap is

thus bounded by $\frac{1}{\beta} \log |\mathcal{F}|$, where $|\mathcal{F}|$ represents the size of \mathcal{F} . This is a direct consequence of us theoretically approximating the max function by a log-sum-exp function in (2.13). Practically, adding this additional entropy term in fact opens new design space for exploration. Second, the approximation becomes exact as β approaches infinity. As discussed in [18], however, usually β should not take too large values as there are practical constraints or convergence rate concerns in the algorithm design afterwards. Third, we can derive a close form of the optimal solution of the problem in (2.16). Let λ be the Lagrange multiplier associated with the equality constraint in (2.16) and $p_f^*(x)$, $f \in \mathcal{F}$ be the optimal solution of the problem in (2.16). By solving the Karush–Kuhn–Tucker (KKT) conditions [19] of the problem in (2.16):

$$W_f - \frac{1}{\beta} \log p_f^*(W) - \frac{1}{\beta} + \lambda = 0, \quad \forall f \in \mathcal{F}, \quad (2.17)$$

$$\sum_{f \in \mathcal{F}} p_f^*(W) = 1, \quad \lambda \geq 0, \quad (2.18)$$

we have

$$p_f^*(W) = \frac{\exp(\beta W_f)}{\sum_{f' \in \mathcal{F}} \exp(\beta W_{f'})}, \quad \forall f \in \mathcal{F}. \quad (2.19)$$

By time-sharing among different configurations f according to $p_f^*(W)$, we can solve the problem **MWC – EQ**, and hence the problem **MWC**, approximately. We remark that the optimality gap is bounded by $\frac{1}{\beta} \log |\mathcal{F}|$, which can be made small by choosing large β .

2.3.1.3 Distributed Markov Chain Monte Carlo

We design a Markov chain with a state space being the set of all feasible peering configurations \mathcal{F} and has a stationary distribution as $p_f^*(W)$ in (2.19). We implement the Markov chain to guide the system to optimize the configuration. As the system hops among configurations, the Markov chain converges and the configurations are time-shared according to the desired distribution $p_f^*(W)$.

The key lies in designing such Markov chain that allows distributed implementation. Since $p_f^*(W)$ in (2.19) is product-form, it suffices to focus on designing time-reversible Markov chains [18].

Let $f, f' \in \mathcal{F}$ be two states of Markov chain, and denote $q_{f,f'}$ as the transition rate from state f to f' . We have two degrees of freedom in designing a time-reversible Markov chain:

- *The state space structure:* we can add or cut direct transitions between any two states, given that the state space remains connected and any two states are reachable from each other. For example, assume that the four-states Markov

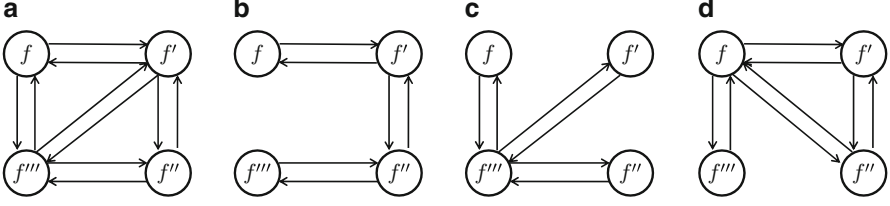


Fig. 2.3 The Markov chains in (b), (c), (d), by adding/removing transition edge-pair between two states in the time-reversible Markov chain in (a), are also time-reversible. All Markov chains have the same stationary distribution

chain in Fig. 2.3a is time-reversible. The “sparse” Markov chains in Fig. 2.3b–d, modified from the “dense” one in Fig. 2.3a by adding/removing transition edge-pair between two states, are also time-reversible. Furthermore, all Markov chains share the same stationary distribution.

- *The transition rates:* we can explore various options in designing $q_{f,f'}$, given that the detailed balance equation is satisfied, i.e.,

$$p_f^*(W)q_{f,f'} = p_{f'}^*(W)q_{f',f}, \forall f, f' \in \mathcal{F}. \quad (2.20)$$

Satisfying the above equations guarantees the designed Markov chain and has the desired stationary distribution as in (2.19). For instance, 802.11b wireless CSMA MAC protocol implements a time-reversible Markov chain with

$$q_{f,f'} = \alpha \exp(\beta(W_{f'} - W_f)), \quad (2.21)$$

$$q_{f',f} = \alpha, \quad (2.22)$$

where α is a positive constant [18]. We show another example on P2P streaming later in this chapter.

In Markov approximation framework, with large values of β , the system hops toward better configurations more greedily. However, this may as well lead to the system getting trapped in locally optimal configurations. Hence there is a trade-off to consider when setting the value of β . The value of β also affects the convergence rate of the time-reversible Markov chain to the desired stationary distribution. It is worth future investigation to further understand the impact of β .

2.3.2 Application to Distributed P2P Streaming

In this subsection, we apply Markov approximation technique elaborated above to design distributed algorithms for maximizing P2P streaming rate under node degree bounds.

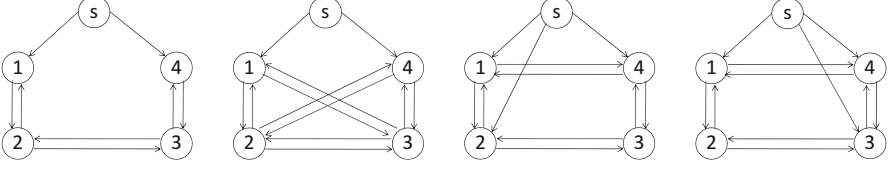


Fig. 2.4 Peering configuration examples for a five-node network with node degree bound 3 for each node

2.3.2.1 Settings and Notations

Like in Sect. 2.2, we model the P2P overlay network as a general directed graph $G = (V, E)$, where V denotes the set of nodes and E denotes the set of links. Each link in the graph corresponds to a TCP/UDP connection between two nodes. Each node $v \in V$ is associated with an upload capacity $C(v) \geq 0$. We assume there is no constraint on the downloading rate for each node $v \in V$. This assumption can be partly justified by the empirical observation that as residential broadband connections with asymmetric upload and download rates become increasingly dominant, bottlenecks typically are at the uplinks of the access networks rather than on end-user download speeds or in the middle of the Internet.

We again focus on the single-source streaming scenario, i.e., a source s broadcasts a continuous stream of contents to the entire network; we denote its receiver set as $R \triangleq V - \{s\}$.

We consider the peering constraints that each node has a degree bound B_v , i.e., it can only *simultaneously* connect to a B_v number of neighbors due to connection overhead cost. We allow different nodes to have different degree bounds. Figure 2.4 shows four sample peering configurations of a five-node network with node degree bound 3 for each node.

2.3.2.2 Problem Formulation and Our Approach

Let \mathcal{F} denote the set of all feasible peering configurations over graph G under node degree bounds. For a configuration $f \in \mathcal{F}$, let x_f be the maximum achievable broadcast rate under f . The problem of maximizing broadcast rate under node degree bounds can be formulated as follows:

$$\text{MRC} : \max_{f \in \mathcal{F}} x_f. \quad (2.23)$$

This problem is known to be NP-complete [2]. To address this problem in a distributed manner, we first develop a distributed broadcasting algorithm that can achieve x_f under arbitrary $f \in \mathcal{F}$. We then design a distributed topology-hopping algorithm that optimizes toward the best peering configurations. They operate in

tandem to achieve a close-to-optimal broadcast rate under arbitrary node degree bounds, and over arbitrary overlay graph. We elaborate on the topology-hopping algorithm in the following. Details of the broadcasting algorithm are in [3].

Following Markov approximation framework, we need to design a Markov chain with a state space being the set of feasible configurations \mathcal{F} and with a stationary distribution as follows:

$$p_f^*(x) = \frac{\exp(\beta x_f)}{\sum_{f' \in \mathcal{F}} \exp(\beta x_{f'})}, \forall f \in \mathcal{F}. \quad (2.24)$$

Since $p_f^*(x)$ in (2.24) is product-form, it suffices to focus on designing time-reversible Markov chains. The key lies in designing such Markov chain that allows distributed implementation.

Let $f, f' \in \mathcal{F}$ be two states of Markov chain, and denote $q_{f,f'}$ as the transition rate from state f to f' . In our Markov chain design, we first specify its state space structure as follows: we set the transition rate $q_{f,f'}$ to be zero, unless f and f' satisfy that

- $|\mathcal{N}_f \cup \mathcal{N}_{f'} - \mathcal{N}_f \cap \mathcal{N}_{f'}| = 2$, i.e., f and f' differ by only two node pairs.
- There exists a node, denoted by v^* , so that $\mathcal{N}_f \cup \mathcal{N}_{f'} - \mathcal{N}_f \cap \mathcal{N}_{f'} \subseteq \{\{v^*, u\}, \forall u \in N_{v^*}\}$. That is, these two node pairs share a common node v^* .

In other words, we only allow direct transitions between two configurations if such transitions correspond to a single node swapping an in-use neighbor with a not-in-use one. Second, given the state space structure of Markov chain, we design the transition rates to favor distributed implementation while satisfying the detailed balance equation.

One possible option is to set $q_{f,f'}$ to be $\exp^{-1}(\beta x_{f'})$. One way to implement this option is for every node to generate a timer according to its *measured* receiving rate and counts down accordingly. When the timer expires, the dedicated node performs the neighbor swapping and resets its timer. As simple as the implementation may sound, this option is expensive to implement. Once the peering configuration changes, the system needs to notify all the nodes to measure the new receiving rate and reset their timers accordingly. It is not clear how to implement such system-wide notification in a low-overhead manner.

Instead, we choose to design $q_{f,f'}$ and $q_{f',f}$ as follows:

$$q_{f,f'} = \frac{1}{\exp(\tau)} \frac{\exp(\beta x_{f'})}{\exp(\beta x_{f'}) + \exp(\beta x_f)} \quad (2.25)$$

and

$$q_{f',f} = \frac{1}{\exp(\tau)} \frac{\exp(\beta x_f)}{\exp(\beta x_f) + \exp(\beta x_{f'})}, \quad (2.26)$$

where τ is a constant. It is straightforward to verify that detailed balance equation is satisfied. Such choices of transition rates do not require coordination or notification among peers in its implementation. We leave details of the distributed implementation to [3].

2.3.2.3 Impact of Inaccurate Observations

In practice, it is possible to obtain only an inaccurate measurement or estimate of x_f . These inaccuracies root in two sources. One is the noisy measurements of the maximum broadcast rates given the configuration. The other is the fast state transition of Markov chain, i.e., the Markov chain transits before the underlying broadcasting algorithm converges and thus it transits based on inaccurate observations of the broadcast rates.

Consequently, the topology hopping Markov chain may *not* converge to the desired stationary distribution $p_f^*(x)$. This observation motivates our following study on the convergence of Markov chain in the presence of inaccurate transition rates.

For each configuration $f \in \mathcal{F}$ with broadcast rate x_f , we assume its corresponding inaccurate observed rate belongs to the bounded region $[x_f - \Delta_f, x_f + \Delta_f]$, following arbitrary distribution. Δ_f is the inaccuracy bound and can be different for different f .

With such random inaccurate observed rates, it turns out that the topology-hopping process is still a time-reversible Markov chain [3]. We denote $\bar{p} : [\bar{p}_f(x), f \in \mathcal{F}]$ as its stationary distribution of the configurations in this Markov chain. Recall that the stationary distribution of the configurations for the original inaccuracy-free topology hopping Markov chain is $p^* : [p_f^*(x), f \in \mathcal{F}]$. We use the total variance distance [20] to quantify the difference between p^* and \bar{p} , as

$$d_{TV}(p^*, \bar{p}) \triangleq \frac{1}{2} \sum_{f \in \mathcal{F}} |p_f^* - \bar{p}_f| \quad (2.27)$$

We have the following main result:

Theorem 2.5. *Let $\Delta_{\max} = \max_{f \in \mathcal{F}} \Delta_f$, and $x_{\max} = \max_{f \in \mathcal{F}} x_f$. The $d_{TV}(p^*, \bar{p})$ are bounded as follows:*

$$0 \leq d_{TV}(p^*, \bar{p}) \leq 1 - \exp(-2\beta\Delta_{\max}). \quad (2.28)$$

*Further, the optimality gap in broadcast rates $|p^*x^T - \bar{p}x^T|$ is bounded as below:*

$$|p^*x^T - \bar{p}x^T| \leq 2x_{\max}(1 - \exp(-2\beta\Delta_{\max})). \quad (2.29)$$

The upper bound on $d_{TV}(p^*, \bar{p})$ shown in (2.28) is general, as it is independent of the number of configurations $|\mathcal{F}|$ and the distributions of inaccurate observed rates. The upper bound on $d_{TV}(p^*, \bar{p})$ shown in (2.28) decreases exponentially with the worst inaccuracy bound Δ_{\max} decreasing. It would be interesting to explore a tighter upper bound on $d_{TV}(p^*, \bar{p})$ than the one in (2.28).

2.4 Distributed Robust Optimization

2.4.1 General Framework

Despite the importance and success of using optimization theory to study communication and network problems, most work in this area makes the unrealistic assumption that the data defining the constraints and objective function of the optimization problem can be obtained precisely. We call the corresponding problems “nominal.” In many practical problems, these data are typically inaccurate, uncertain, or time-varying. Solving the nominal optimization problems may lead to poor or even infeasible solutions when deployed.

Over the last 10 years, robust optimization has emerged as a framework of tackling optimization problems under data uncertainty (e.g., [21–25]). The basic idea of robust optimization is to seek a solution which remains feasible and near-optimal under the perturbation of parameters in the optimization problem. Each robust optimization problem is defined by three-tuple: *a nominal formulation*, *a definition of robustness*, and *a representation of the uncertainty set*. The process of making an optimization formulation robust can be viewed as a mapping from one optimization problem to another. A central question is as follows: when will important properties, such as convexity and decomposability, be preserved under such mapping? In particular, what kind of nominal formulation and uncertainty set representation will preserve convexity and decomposability in the robust version of the optimization problem?

So far, almost all of the work on robust optimization focuses on determining what representations of data uncertainty preserves convexity, thus tractability through a centralized solution, in the robust counter part of the nominal problem for a given definition of robustness. For example, for the worst-case robustness, it has been shown that under the assumption of ellipsoid set of data uncertainty, a robust linear optimization problem can be converted into a second-order cone problem; and a robust second-order cone problem can be reformulated as a semi-definite optimization problem [26].

In this section, motivated by needs in communication networking, we will focus instead on the *distributiveness*-preserving formulation of the robust optimization. The driving question thus becomes: how much more communication overhead is introduced in making the problem robust?

To develop a systematic theory of distributed robust optimization (DRO), we first show how to represent an uncertainty set in a way that not only captures the data uncertainty in the model but also leads to a distributively solvable optimization problem. Second, in the case where a fully distributed algorithm is not obtainable, we focus on the *tradeoff* between robustness and distributiveness. Distributed algorithms are often developed based on decomposability structure of the problem, which may disappear as the optimization formulation is made robust. While distributed computation has long been studied, unlike convexity of a problem, distributiveness of an algorithm does not have a widely agreed definition. It is often

quantified by the amount of communication overhead required: how far and how frequent do the nodes have to pass message around? Zero communication overhead is obviously the “most distributed”, and we will see how the amount of overhead trades-off with the degree of robustness.

There are many uncertainties in the design of communication networks. These uncertainties stem from various sources and can be broadly grouped into two categories. The first type of uncertainties are related to the perturbation of a set of design parameters due to erroneous inputs such as errors in estimation or implementation. We call them *perturbation errors*. A common characteristic of such perturbation errors is that they can often be modeled as a *continuous* uncertainty set surrounding the basic point estimate of the parameter. The size of the uncertainty set could be used to characterize the level of perturbations the designer needs to protect against. The second type of uncertainties is termed as *disruptive errors*, as they are caused by the failure of communication links within the network. This type of errors can be modeled as a *discrete* uncertainty set.

As a starter, we will focus on a class of optimization problems with the following nominal form: maximization of a *concave* objective function over a given data set characterized by *linear* constraints,

$$\begin{aligned} & \text{maximize } f_0(x) \\ & \text{subject to } Ax \preceq b \\ & \text{variables } x, \end{aligned} \tag{2.30}$$

where A is an $M \times N$ matrix, x is an $N \times 1$ vector, and b is an $M \times 1$ vector. This class of problems can model a wide range of engineering systems (e.g., [27]).

The uncertainty of problem (2.30) may exist in the objective function f_0 , matrix parameter A , and vector parameter b . In many cases, the uncertainty in objective function f_0 can be converted into uncertainty of the parameters defining the constraints [28]. It is also possible to convert the uncertainty in b into uncertainty in A in certain cases (although this could be difficult in general). In the rest of this section, we will focus on studying the uncertainty in A . In many networking problems, structures and physical meaning of matrix A readily lead to distributed algorithms, e.g., in rate control where A is a given routing matrix, distributed subgradient algorithm is well known to solve the problem, with an interesting correspondence with the practical protocol of TCP. Making the optimization robust may turn the linear constraints into nonlinear ones and increase the amount of message passing.

In the robust counterpart of problem (2.30), we require the constraints $Ax \preceq b$ to be valid for any $A \in \mathcal{A}$, where \mathcal{A} denotes the uncertainty set of A , and the definition of robustness is in the *worst-case* sense [19]. This approach might be too conservative. A more meaningful choice of robustness is the *chance-constrained robustness*, i.e., the probability of infeasibility (or outage) is upper bounded. We can flexibly adjust the chance-constrained robustness of the robust solution by solving the worst-case robust optimization problem over a properly selected subset of the exact uncertainty set.

If we allow an arbitrary uncertainty set A , then the robust optimization problem is difficult to solve even in a centralized manner [29]. We will focus on the study of *constraint-wise* (i.e., *row-wise*) uncertainty set, where the uncertainties between different rows in matrix A are decoupled. This restricted class of uncertainty set characterizes the data uncertainty in many practical problems, and it also allows us to convert the robust optimization problem into a formulation that is distributively solvable.

Denote the j^{th} row of A be a_j^T , which lies in a compact uncertainty set \mathcal{A}_j . Then the *robust* optimization problem that we focus on in this chapter can be written in the following form:

$$\begin{aligned} & \text{maximize } f_0(x), \\ & \text{subject to } a_j^T x \leq b_j, \forall a_j \in \mathcal{A}_j, \forall 1 \leq j \leq M, \\ & \text{variables } x. \end{aligned} \tag{2.31}$$

It is easy to see that the robust optimization problem (2.31) can be equivalently written in a form represented by *protection functions* instead of uncertainty sets. Denote the nominal counterpart of problem (2.31) with a coefficient matrix \bar{A} (i.e., the values when there is no uncertainty), with the j th row's coefficient $\bar{a}_j \in \mathcal{A}_j$. Then

Theorem 2.6. *Problem (2.31) is equivalent to the following convex optimization problem:*

$$\begin{aligned} & \text{maximize } f_0(x), \\ & \text{subject to } \bar{a}_j^T x + g_j(x) \leq b_j, \forall 1 \leq j \leq M, \\ & \text{variables } x, \end{aligned} \tag{2.32}$$

where

$$g_j(x) = \max_{a_j \in \mathcal{A}_j} (a_j - \bar{a}_j)^T x, \tag{2.33}$$

is the protection function for the j th constraint, which depends on the uncertainty set \mathcal{A}_j and the nominal row \bar{a}_j . Furthermore, $g_j(x)$ is a convex function since for any $0 \leq t \leq 1$, we have that

$$\max_{a_j \in \mathcal{A}_j} (a_j - \bar{a}_j)^T [tx_1 + (1-t)x_2] \leq t \max_{a_j \in \mathcal{A}_j} (a_j - \bar{a}_j)^T x_1 + (1-t) \max_{a_j \in \mathcal{A}_j} (a_j - \bar{a}_j)^T x_2. \tag{2.34}$$

Different forms of \mathcal{A}_j will lead to different protection function $g_j(x)$, which results in different robustness and performance tradeoff of the formulation. Next we consider several approaches in terms of modeling \mathcal{A}_j and the corresponding protection function $g_j(x)$.

2.4.1.1 Polyhedron Uncertain Set

In this case, the uncertainty set \mathcal{A}_j is a polyhedron characterized by a set of linear inequalities, i.e., $\mathcal{A}_j \triangleq \{a_j : D_j a_j \preceq c_j\}$. The protection function is

$$g_j(x) = \max_{a_j: D_j a_j \preceq c_j} (a_j - \bar{a}_j)^T x, \quad (2.35)$$

which involves a linear program (LP). We next show that the uncertainty set can be translated into a set of linear constraints. In the j th constraint in (2.31), with $x = \hat{x}$ fixed, we can characterize the set $\forall a_j \in \mathcal{A}_j$ by comparing b_j with the outcome of the following LP:

$$v_j^* = \max_{a_j: D_j a_j \preceq c_j} a_j^T \hat{x}. \quad (2.36)$$

If $v_j^* \leq b_j$, then \hat{x} is feasible for (2.31). However, this approach is not very useful since it requires solving one LP in (2.36) for each possible \hat{x} . Alternatively, we take the Lagrange dual problem of the LP in (2.36),

$$v_j^* = \min_{p_j: D_j^T p_j \succeq \hat{x}, p_j \succeq 0} c_j^T p_j. \quad (2.37)$$

If we can find a feasible solution \hat{p}_j for (2.37), and $c_j^T \hat{p}_j \leq b_j$, then we must have $v_j^* \leq c_j^T \hat{p}_j \leq b_j$. We can thus replace constraints in (2.31) by the following constraints:

$$c_j^T p_j \leq b_j, D_j^T p_j \succeq x, p_j \succeq 0, \forall 1 \leq j \leq M, \quad (2.38)$$

and we now have an equivalent and *deterministic* formulation for problem (2.31), where all the constraints are linear.

2.4.1.2 D-Norm Uncertainty Set

D -norm approach [28] is another method to model the uncertainty set, and has advantages such as guarantee of feasibility independent of uncertainty distributions and flexibility in trading off between robustness and performance.

Consider the j th constraint $a_j^T x \leq b_j$ in (2.31). Denote the set of all uncertain coefficients in a_j as \mathcal{E}_j . The size of \mathcal{E}_j is $|\mathcal{E}_j|$, which might be smaller than the total number of coefficients N (i.e., a_{ij} for some i might not have uncertainty). For each $a_{ij} \in \mathcal{E}_j$, assume the actual value falls into the range of $[\bar{a}_{ij} - \hat{a}_{ij}, \bar{a}_{ij} + \hat{a}_{ij}]$, in which \hat{a}_{ij} is a given error bound. Also choose a nonnegative integer $\Gamma_j \leq |\mathcal{E}_j|$. The definition of robustness associated with the D -norm formulation is to maintain feasibility if at most Γ_j out of all possible $|\mathcal{E}_j|$ parameters are perturbed. Let us denote \mathcal{S}_i as the set of Γ_j uncertain coefficients. As is well known, the above

robustness definition can be characterized by the following protection function,

$$g_j(x) = \max_{S_j: S_j \subseteq \mathcal{E}_j, |S_j| = \Gamma_j} \sum_{i \in S_j} \hat{a}_{ij} |x_i|. \quad (2.39)$$

If $\Gamma_j = 0$, then $g_j(\Gamma_j, x) = 0$ and the j th constraint is reduced to the nominal constraint. If $\Gamma_j = |\mathcal{E}_j|$, then $g_j(x) = \sum_{i \in \mathcal{E}_j} \hat{a}_{ij} |x_i|$, and the j th constraint becomes Soyster's worst-case formulation [28]. The tradeoff between robustness and performance can be obtained by adjusting Γ_j .

2.4.1.3 Ellipsoid Uncertainty Set

Ellipsoid is commonly used to approximate complicated uncertainty sets for statistical reasons [29] and for its ability to succinctly describe a set of discrete points in Euclidean geometry [19]. Here we consider the case where coefficient a_j falls in an ellipsoid centered at the nominal \bar{a}_j . Specifically,

$$\mathcal{A}_j = \{\bar{a}_j + \Delta a_j : \sum_i |\Delta a_{ij}|^2 \leq \varepsilon_j^2\}. \quad (2.40)$$

By (2.33), the protection function is given by

$$g_j(x) = \max \left\{ \sum_i \Delta a_{ij} x_i : \sum_i |\Delta a_{ij}|^2 \leq \varepsilon_j^2 \right\}, \quad (2.41)$$

Denote by $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$ as the ℓ_2 -norm (or the Euclidean norm) of x . By Cauchy–Schwartz inequality,

$$\sum_i \Delta a_{ij} x_i \leq \sqrt{\sum_i |\Delta a_{ij}|^2} \|x\|_2 \leq \varepsilon_j \|x\|_2,$$

and the equality is attained by choosing

$$\Delta a_{ij} = \frac{x_i \varepsilon_j}{\|x\|_2}.$$

Therefore, we conclude that

$$g_j(x) = \varepsilon_j \|x\|_2. \quad (2.42)$$

2.4.2 Robust Multipath Rate Control for Network Service Reliability

2.4.2.1 Nominal and Robust Formulations

Consider a wireline network where some links might fail because of human mistakes, software bugs, hardware defects, or natural hazard. Network operators typically reserve some bandwidth for some backup paths. When the primary paths fail, some or all of the traffic will be rerouted to the corresponding disjoint backup paths. Fast system recovery schemes are essential to ensure service availability in the presence of link failure. There are three key components for fast system recovery [30]: identifying a backup path disjoint from the primary path, computing network resource (such as bandwidth) in reservation prior to link failure, and detecting the link failure in real-time and rerouting the traffic. The first component has been investigated extensively in graph theory. The third component has been extensively studied in system research community. Here we consider the robust rate control and bandwidth reservation in the face of possible failure of primary path, which is related to the second component.

We first consider the nominal problem with no link failures. Following similar notation as in Kelly's seminal work [27], we consider a network with S users, L links, and T paths, indexed by s , l , and t , respectively. Each user is a unique flow from one source node to one destination node. There could be multiple users between the same source–destination node pair. The network is characterized by the $L \times T$ path-availability 0 – 1 matrix

$$[D]_{lt} = \begin{cases} d_{lt} = 1, & \text{if link } l \text{ is on path } t, \\ 0, & \text{otherwise.} \end{cases}$$

and $T \times S$ primary-path-choice nonnegative matrix

$$[W]_{ts} = \begin{cases} w_{ts}, & \text{if user } s \text{ uses path } t \text{ as the primary path,} \\ 0, & \text{otherwise.} \end{cases}$$

where $w_{ts} > 0$ indicates the percentage that user s allocates its rate to primary path t , and $\sum_t w_{ts} = 1$. Let x , c , and y denote source rates, link capacities, and aggregated path rates, respectively. The nominal multipath rate control problem is

$$\begin{aligned} & \text{maximize } \sum_s f_s(x_s) \\ & \text{subject to } Dy \preceq c, \quad Wx \preceq y, \\ & \text{variables } x \succeq 0, y \succeq 0, \end{aligned} \tag{2.43}$$

where $f_s(x_s)$ is user s ' utility function, which is increasing and strictly concave in x_s .

To represent Problem (2.43) in a more compact way, we denote $R = DW$ as the link-source matrix

$$[R]_{ls} = \begin{cases} r_{ls} = \sum_{t \in T(l)} w_{ts}, & \text{if user } s \text{ uses link } l \text{ in one of its primary paths,} \\ 0, & \text{otherwise.} \end{cases}$$

where $T(l)$ denotes the set of all paths associated with link l , i.e., $T(l) = \{t : d_{lt} = 1\}$.

The nominal problem can be compactly rewritten as

$$\begin{aligned} & \text{maximize} && \sum_s f_s(x_s) \\ & \text{subject to} && Rx \preceq c, \\ & \text{variables} && x \succeq 0. \end{aligned} \tag{2.44}$$

To ensure robust data transmission against the link failures, each user also determines a backup path when it joins the network. The nonnegative backup path choice matrix is

$$[B]_{ts} = \begin{cases} b_{ts}, & \text{if user } s \text{ uses path } t \text{ as the backup path,} \\ 0, & \text{otherwise.} \end{cases}$$

where $b_{ts} > 0$ indicates the maximum percentage that user s allocates its rate to path t . The actual rate allocation will be a random variable between 0 and b_{ts} , depending on whether the primary paths fail. We further assume that a path can only be used as either a primary path or a backup path for the same user but not both. The corresponding *robust multipath routing rate allocation problem* is given by

$$\begin{aligned} & \text{maximize} && \sum_s f_s(x_s) \\ & \text{subject to} && \sum_{s \in S(l)} r_{ls} x_s + \sum_{t \in T(l)} g_t(b_t, x) \leq c_l, \quad \forall l. \\ & \text{variables} && x \succeq 0. \end{aligned} \tag{2.45}$$

Here $S(l)$ denotes the set of users using the link l in one of their primary paths, and $\sum_{s \in S(l)} r_{ls} x_s$ is the aggregate rate from users. Moreover, $g_t(b_t, x)$ is the protection function for the traffic from users who use path t as their backup path, and b_t is the t th row of matrix B .

There are several ways of characterizing the protection function. Here we take D -norm as an example. Let $\mathcal{E}_t = \{s : b_{ts} > 0, \forall s\}$ denote the set of users who utilize path t as the backup path, and $\mathcal{F}_{t, \Gamma_t}$ denote a set such that

$$\mathcal{F}_{t, \Gamma_t} \subseteq \mathcal{E}_t \text{ and } |\mathcal{F}_{t, \Gamma_t}| = \Gamma_t.$$

Here Γ_t denotes the number of users who might experience path failures, and its value controls the tradeoff between robustness and performance. The protection function can then be written as

$$g_t(b_t, x) = \max_{\mathcal{F}_t, \Gamma_t \subseteq \mathcal{E}_t} \sum_{s \in \mathcal{F}_t, \Gamma_t} b_{ts} x_s, \forall t. \quad (2.46)$$

Notice that Γ_t is a parameter (instead of a variable) in the protection function (2.46).

A centralized algorithm such as the interior point method can be used to solve the service reliability problem (2.45). In practice, however, a distributed solution is preferred in many situations. For example, for a large-scale communication network, multiple nodes and associated links might be removed or updated, and it could be difficult to collect the updated topology information of the whole network. Furthermore, a distributed algorithm can be used to dynamically adjust the rates according to changes in the network topology. Since we consider service reliability in this application, most likely such a update is only required infrequently.

2.4.2.2 Distributed Primal–Dual Algorithms

We now develop a fast distributed algorithm to solve the robust optimization of multipath rate control based on a combination of active-set method [31] and dual-based decomposition method.

We first show that the nonlinear constraints in Problem (2.45) can be replaced by a set of linear constraints:

Proposition 2.1. *For any path t , the single constraint*

$$\sum_{s \in S(l)} r_{ls} x_s + \sum_{t \in T(l)} g_t(b_t, x) \leq c_l, \quad (2.47)$$

is equivalent to the following set of constraints

$$\sum_{s \in S(l)} r_{ls} x_s + \sum_{t \in T(l)} \sum_{s \in \mathcal{F}_t, \Gamma_t} b_{ts} x_s \leq c_l, \forall (\mathcal{F}_t, \Gamma_t, \forall t \in T(l)) \text{ such that } \mathcal{F}_t, \Gamma_t \subseteq \mathcal{E}_t. \quad (2.48)$$

We further define some short-hand notation. For each choice of parameters ($|T(l)|$ sets of users)

$$\mathcal{F}_l = (\mathcal{F}_t, \Gamma_t \subseteq \mathcal{E}_t, \forall t \in T(l)),$$

we define a set corresponding to the choices of paths and users:

$$\mathcal{Q}_{\mathcal{F}_l} \triangleq \{(t, s) : t \in T(l), s \in \mathcal{F}_t, \Gamma_t\}.$$

We further define $\mathcal{G}_l = \{\mathcal{Q}_{\mathcal{F}_l}, \forall \mathcal{F}_l\}$. The constraints in (2.48) can be rewritten as follows:

$$\sum_{s \in S(l)} r_{ls} x_s + \sum_{(t,s) \in \mathcal{Q}_{\mathcal{F}_l}} b_{ts} x_s \leq c_l, \quad \forall \mathcal{Q}_{\mathcal{F}_l} \in \mathcal{G}_l. \quad (2.49)$$

Note the number of constraints in (2.49) is $\prod_{l \in T(l)} \binom{|\mathcal{E}_l|}{|\Gamma_l|}$, and increases quickly with Γ_l and $|\mathcal{E}_l|$. This motivates us to design an alternative method to solve (2.45). We next present a sequential optimization approach. The basic idea is to iteratively generate a set $\bar{\mathcal{G}}_l \subseteq \mathcal{G}_l$, and use the following set of constraints to approximate (2.49):

$$\sum_{s \in S(l)} r_{ls} x_s + \sum_{(t,s) \in \mathcal{Q}_{\mathcal{F}_l}} b_{ts} x_s \leq c_l, \quad \forall \mathcal{Q}_{\mathcal{F}_l} \in \bar{\mathcal{G}}_l. \quad (2.50)$$

This leads to a relaxation of Problem (2.45):

$$\begin{aligned} & \text{maximize} \quad \sum_s f_s(x_s) \\ & \text{subject to} \quad \sum_{s \in S(l)} r_{ls} x_s + \sum_{(t,s) \in \mathcal{Q}_{\mathcal{F}_l}} b_{ts} x_s \leq c_l, \quad \forall \mathcal{Q}_{\mathcal{F}_l} \in \bar{\mathcal{G}}_l, \quad \forall l, \\ & \text{variables } x \succeq 0. \end{aligned} \quad (2.51)$$

Let \bar{x} denote an optimal solution to the relaxed problem (2.51) and x^* denote an optimal solution of the original problem (2.45). If $\bar{\mathcal{G}}_l = \mathcal{G}_l$, then we have $\sum_s f_s(\bar{x}_s) = \sum_s f_s(x_s^*)$. Even if $\bar{\mathcal{G}}_l \subset \mathcal{G}_l$, it is still possible that the two optimal objective values are the same.

Proposition 2.2. $\sum_s f_s(\bar{x}_s) = \sum_s f_s(x_s^*)$ if the following condition holds

$$\max_{\mathcal{Q}_{\mathcal{F}_l} \in \bar{\mathcal{G}}_l} \sum_{(t,s) \in \mathcal{Q}_{\mathcal{F}_l}} b_{ts} \bar{x}_s = \max_{\mathcal{Q}_{\mathcal{F}_l} \in \mathcal{G}_l} \sum_{(t,s) \in \mathcal{Q}_{\mathcal{F}_l}} b_{ts} \bar{x}_s, \quad \forall l. \quad (2.52)$$

Next we develop a distributed algorithm to solve Problem (2.51) for a fixed $\bar{\mathcal{G}}_l$ for each l , which is suboptimal for solving Problem (2.45). We can then design an optimal distributed algorithm that achieves the optimal solution of Problem (2.45) by iteratively using the following algorithm.

First, by relaxing the constraints in Problem (2.51) using dual variables $\lambda = \{\{\lambda_{li}\}_{i=1}^{|\bar{\mathcal{G}}_l|}\}_{l=1}^L$, we obtain the following Lagrangian,

$$\bar{Z}(\lambda, x) = \sum_s f_s(x_s) + \sum_l \sum_{i=1}^{|\bar{\mathcal{G}}_l|} \lambda_{li} \left(c_l - \sum_{s \in S(l)} r_{ls} x_s - \sum_{(t,s) \in \mathcal{Q}_{\mathcal{F}_l}(i)} b_{ts} x_s \right).$$

For easy indexing, here we denote each set $\bar{\mathcal{G}}_l = \{\mathcal{Q}_{\mathcal{F}_l}(i), i = 1, \dots, |\bar{\mathcal{G}}_l|\}$.

Distributed Primal-dual Subgradient Algorithm

$k \leftarrow 0, \lambda(0) \leftarrow \mathbf{0}, \mu(0) \leftarrow \mathbf{0}$, and choose $\epsilon \ll 1$

while there exists an i so that $\eta_i(k) = 1 - \frac{\sum_s f_s(\bar{x}_{s,i}(k))}{Z(\lambda_k)} > \epsilon_i$
 $k \leftarrow k + 1$
 Each user s determines $x_s(k)$ by solving Problem (1.54)
 Each user s passes the value of $x_s(k)$ to each link associated with this user
 Each user s who uses link l on either its primary or backup paths calculates
 the associated dual prices by passing messages over path t
 from its destination to its source
 Each user s computes $\bar{\Delta}_{s,i}(k) = \min_l \left[\left(\bar{\Delta}_s(l, k) \right)_{r_{ls} > 0}, \left(\bar{\Delta}_s(l, k) \right)_{b_{ts} > 0, (t,s) \in \mathcal{Q}_{\mathcal{F}_l}(i)} \right]$,
 and $\bar{x}_{s,i}(k) = x_s(k) \bar{\Delta}_{s,i}(k)$ by collecting messages from its associated links
end while

Fig. 2.5 Distributed primal–dual subgradient algorithm for robust optimization of multipath rate control

The dual function is

$$Z(\lambda) = \max_{x \geq 0} \bar{Z}(\lambda, x). \quad (2.53)$$

The optimization over x in (2.53) can be decomposed into one problem for each user s :

$$\max_{x_s \geq 0} \left(f_s(x_s) - \sum_l \sum_{i=1}^{|\bar{\mathcal{G}}_l|} \left(\lambda_{li} r_{ls} + \lambda_{li} \sum_{(t,s) \in \mathcal{Q}_{\mathcal{F}_l}(i)} b_{ts} \right) x_s \right). \quad (2.54)$$

Notice that link l can be viewed as the composition of $|\bar{\mathcal{G}}_l|$ sub-links and each sub-link is associated with a price (dual variable) λ_{li} . Each user s determines its transmission rate x_s by considering prices from both its primary path and backup path.

The master dual problem is

$$\min_{\lambda \geq 0} Z(\lambda), \quad (2.55)$$

which can be solved by the subgradient method. For each dual variable λ_{li} , its subgradient can be calculated as

$$\zeta_{li}(\lambda_{li}) = c_l - \sum_{s \in S(l)} r_{ls} x_s - \sum_{(t,s) \in \mathcal{Q}_{\mathcal{F}_l}(i)} b_{ts} x_s. \quad (2.56)$$

The value of λ_{li} will be updated using the subgradient information correspondingly.

Pseudo-code of the algorithm is in Fig. 2.5. Here $\theta(k)$ is the step-size at time k . If it meets conditions such as those given in [32, p. 505], then this subgradient method is guaranteed to converge to the optimal solution. Common choices of step

size include the constant step size ($\theta(k) = \beta$), diminish step size ($\theta(k) = \frac{\beta}{\sqrt{k}}$), and square summable ($\theta(k) = \frac{\beta}{k}$). The parameter ε_i is a predefined small threshold.

Due to the special structure of (2.51), we can characterize the suboptimality gap of the obtained solution at each iteration, as shown below.

Proposition 2.3. *Let $P^*(\{\bar{\mathcal{G}}_l\})$ be the optimal object function value of the relaxed problem (2.51), λ^k be the dual variables at the k^{th} iteration of the algorithm above, and $[\bar{x}_1(k), \dots, \bar{x}_S(k)]$ be a feasible solution to (2.51). Then $P^*(\{\bar{\mathcal{G}}_l\})$ is lower bounded by $\sum_s f_s(\bar{x}_s(k))$ and upper bounded by $Z(\lambda^k)$, i.e.,*

$$\sum_s f_s(\bar{x}_s(k)) \leq P^*(\{\bar{\mathcal{G}}_l\}) \leq Z(\lambda^k). \quad (2.57)$$

Furthermore, the suboptimality gap $\eta_i(k)$ satisfies $\lim_{k \rightarrow \infty} \eta_i(k) = 0$ for all i .

2.5 Summary

To summarize, these applications to rate control and P2P capacity are only small samples of the use of distributed optimization in networking. This chapter serves as a terse teaser. Much more on existing results, including proofs and numerical examples, can be found from the papers referenced here. Much more also remains to be explored as the research community continues to formulate difficult and robust optimization problems and try out distributed solutions.

Acknowledgements Minghua Chen was partially supported by the China 973 Program (Project No. 2012CB315904), and the General Research Fund grants (Project Nos. 411209, 411010, and 411011) and an Area of Excellence Grant (Project No. AoE/E-02/08), established under the University Grant Committee of the Hong Kong SAR, China, as well as two gift grants from Microsoft and Cisco. Mung Chiang was partially supported by AFOSR MURI grant FA9550-09-1-0643.

References

1. Sengupta, S., Liu, S., Chen, M., Chiang, M., Li, J., Chou, P.: Peer-to-peer streaming capacity. *IEEE Trans. Inform. Theory* **57**(8), 5072–5087 (2011)
2. Liu, S., Chen, M., Sengupta, S., Chiang, M., Li, J., Chou, P.: “Peer-to-Peer Streaming Capacity under Node Degree Bound”, *Proceedings of the 30th International Conference on Distributed Computing Systems (ICDCS 2010)*, Genoa, Italy, 21–25 (2010)
3. Zhang, S., Shao, Z., Chen, M.: Optimal distributed p2p streaming under node degree bounds. In: *Proceedings of the IEEE ICNP, Kyoto, Japan* (2010)
4. Garg, N., Konemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science (FOCS 1998)*, Palo Alto, CA, USA, 8–11 (1998)

5. Dantzig, G., Wolfe, P.: Decomposition principle for linear programs. *Oper. Res.* **8**, 101–111 (1960)
6. Gilmore, P., Gomory, R.: A linear programming approach to the cutting-stock problem. *Oper. Res.* **9**, 849–859 (1961)
7. Gilmore, P., Gomory, R.: A linear programming approach to the cutting stock problem—Part II. *Oper. Res.* **11**, 863–888 (1963)
8. Lin, X., Shroff, N., Srikant, R.: A tutorial on cross-layer optimization in wireless networks. *IEEE J. Sel. Areas Commun.* **24**(8), 1452–1463 (2006)
9. Jiang, L., Walrand, J.: “A distributed csma algorithm for throughput and utility maximization in wireless networks”. In: *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing*, Urbana, IL, USA (2008)
10. Chen, M., Liew, S., Shao, Z., Kai, C.: Markov approximation for combinatorial network optimization. CUHK Technical Report (2009). Available at <http://www.ie.cuhk.edu/~mhchen/ma.tr.pdf>
11. Zhang, H., Chen, M., Parekh, A., Ramchandran, K.: An adaptive multi-channel p2p video-on-demand system using plug-and-play helpers. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-111, July 2010. [Online] Available: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-111.html>
12. Jiang, W., Lan, T., Ha, S., Chen, M., Chiang, M.: “Joint VM Placement and Routing for Data Center Traffic Engineering”. In: *Proceedings of IEEE INFOCOM (mini-conference)*, Orlando, Florida, USA, 25–30 (2012)
13. Chen, M., Liew, S.C., Shao, Z., Kai, C.: Markov approximation for combinatorial network optimization. In: *Proceedings of the IEEE INFOCOM*, San Diego (2010)
14. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic aware virtual machine placement. In: *Proceedings of the IEEE INFOCOM 2010*, San Diego, CA, US (2010)
15. Garg, N., Konjevod, G., Ravi, R.: A polylogarithmic approximation algorithm for the group Steiner tree problem. In the 9th Annual ACM-SIAM SODA (1998), San Francisco, CA, USA, 25–27 (1998)
16. Feige, U.: A threshold of $\ln n$ for approximating set cover. In: *Proceedings of the 28th ACM STOC* (1996), Philadelphia, PA, USA, 22–24 (1996)
17. Charikar, M., Chekuri, C., Cheung, T., Dai, Z., Goel, A., Guha, S., Li, M.: Approximation algorithms for directed Steiner tree problems. In: *Proceedings of the 9th Annual ACM-SIAM SODA* (1998), San Francisco, CA, USA, 25–27 (1998)
18. Chiang, M.: *Geometric programming for communication systems*, Now Publishers Inc (2005)
19. Boyd, S., Vandenberghe, L.: *Convex Optimization*. Cambridge University Press, Cambridge, UK (2004)
20. Diaconis, P., Stroock, D.: Geometric bounds for eigenvalues of Markov chains. *Ann. Appl. Prob.* **1**, 36–61 (1991)
21. Ben-Tal, A., Nemirovski, A.: Robust solutions to uncertain programs. *Oper. Res. Lett.* **25**(1), 1–13 (1999)
22. Ben-Tal, A., Nemirovski, A.: Selected topics in robust convex optimization. *Math. Program.* **112**(1), 125–158 (2008)
23. Nemirovski, A.: On tractable approximations of randomly perturbed convex constraints. In: *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, HI, USA, 8–12 (2003)
24. El-Ghaoui, L.: Robust solutions to least-square problems to uncertain data matrices. *SIAM J. Matrix Anal. Appl.* **18**, 1035–1064 (1997)
25. El Ghaoui, L., Oustry, F., Lebret, H.: Robust solutions to uncertain semidefinite programs. *SIAM J. Optim.* **9**, 33 (1998)
26. Ben-Tal, A., Nemirovski, A.: Robust solutions to linear programming problems contaminated with uncertain data. *Math. Program.* **88**, 411–424 (2000)
27. Kelly, F., Maulloo, A., Tan, D.: Rate control for communication networks: shadow prices, proportional fairness and stability. *J. Oper. Res. Soc.* 237–252 (1998).

28. Bertsimas, D., Sim, M.: The price of robustness. *Oper. Res.* **52**(1), 35–53 (2004)
29. Ben-Tal, A., Nemirovski, A.: Robust solutions of uncertain linear programs. *Oper. Res. Lett.* **25**(1), 1–14 (1999)
30. Xu, D., Li, Y., Chiang, M., Calderbank, A.: Optimal provision of elastic service availability. In *Proceedings of IEEE INFOCOM (2007)*, Anchorage, AK, USA, 6–12 (2007)
31. Leyffer, S.: The return of the active set method. *Oberwolfach Rep.* **2**(1) (2005)
32. Bertsimas, D., Tsitsiklis, J.N.: *Introduction to Linear Optimization*. Athena Scientific, Belmont, Mass., USA (1997)

Modeling and Optimization: Theory and Applications
Selected Contributions from the MOPTA 2010
Conference

Terlaky, T.; Curtis, F.E. (Eds.)

2012, VIII, 120 p., Hardcover

ISBN: 978-1-4614-3923-3