UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a doctoral dissertation by

Ketan Rajawat

and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the final examining committee have been made.

Georgios. B. Giannakis

Name of Faculty Advisor(s)

Signature of Faculty Advisor(s)

Date

GRADUATE SCHOOL

Dynamic Optimization and Monitoring in Communication Networks

A DISSERTATION

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Ketan Rajawat

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Professor Georgios B. Giannakis, Advisor

October 2012

Acknowledgments

I wish to acknowledge and thank everyone who has contributed to this thesis in direct and indirect ways. My deepest gratitude goes to Prof. Georgios B. Giannakis who served not only as my academic advisor but also as my mentor for the last five years. His expansive expertise, patient guidance, and constant encouragement have made my graduate studies rewarding. Due thanks go to Profs. Jarvis Haupt, Nikos Sidiropoulos, and Shuzhong Zhang for agreeing to serve on my committee.

During my time as a graduate student in Minnesota, I greatly benefited from collaborating with Dr. Alfonso Cano, Dr. Emiliano Dall'Anese, Pedro A. Forero, Dr. Seung-Jun Kim, and Nikolaos Gatsis. They deserve not only my gratitude, but also due credit for their contributions to the work reported here. This work also benefited from helpful discussions with current and former members of SPiNCOM: Dr. Daniele Angelosante, Brian Baingana, Juan A. Bazerque, Dr. Shahrokh Farahmand, Dr. Vassilis Kekatos, Prof. Geert Leus, Guobing Li, Dr. Gonzalo Mateos, Morteza Mardani, Prof. Antonio G. Marques, Dr. Eric Msechu, Prof. Alejandro Ribeiro, Prof. Ioannis Schizas, Nasim Yahya Soltani, Dr. Tairan Wang, Hao Zhu, and Yu Zhang. Not only did I receive from them suggestions, ideas, and insights, but also their invaluable friendship. I would also like to thank Prof. Ajit Chaturvedi for encouraging me to further pursue graduate studies.

I am grateful to all my friends here in the States, as well as back in India, for their care and support, and for not letting me feel lonely. My roommate and fellow graduate student Pulkit Jain deserves special thanks for being such good company, and for all the laughs and lunches we shared.

Last, but definitely not the least, my family deserves my heartfelt thanks and appreciation. My parents have given me their unending love and support, without which none of this would have been its worth.

Ketan Rajawat, Minneapolis, Minnesota, November 17, 2012.

Abstract

Communication networks have evolved from specialized, research- and military-oriented transmission systems to large-scale and highly complex interconnections of intelligent devices. Effective operation of such large-scale networks hinges upon real-time allocation of network resources that match the user demands. This thesis contributes towards several key problems encountered in both, monitoring and resource allocation in networks.

Volatile operating environments encountered in ad hoc and sensor networks place severe restrictions on the resources (bandwidth and power) available to network nodes. Pertinent approaches have sought to replicate the Internet protocols in ad hoc networks, exacerbating the resource scarcity by ignoring the peculiarities of the underlying wireless interface. The present thesis leverages the ground-breaking idea of network coding to design wireless network protocols. Towards this end, a cross-layer design is pursued, and network codes are optimized jointly with protocols operating at application, medium access control (MAC), and physical (PHY) layers. For *wireless fading networks*, dual decomposition is utilized to optimally integrate network coding into the protocol stack. Network coding is also introduced for use in Aloha-based MAC, and the resulting non-convex problem is solved via successive convex approximation to realize *practical network coding algorithms*. Benefits of network coding also extend to *QoS-constrained scenarios*, such as in real-time and streaming media applications. Modeling constraints on packet deadlines is the key challenge here, and constant-factor approximations are proposed to this end. In sensor networks where the observed data is correlated across nodes, network coding can both compress and communicate the data to a collection agent. An efficient decoding scheme for this *network-compressive scheme* is developed, yielding network-wide energy savings and increase in the network lifetime.

Exhaustive monitoring of large-scale networks may be challenging or even impossible to perform, motivating the need to account for missing measurements. This thesis puts forth the novel concept of *dynamic network cartography* as tool for inference, tracking, and prediction of the network state. Tapping into the spatio-temporal kriging theory, a dynamic network kriging approach is developed with real-time network-wide prediction capabilities based on latency measurements acquired for a small subset of network paths. Going well beyond state-of-the-art methods, the proposed model captures not only spatio-temporal correlations, but also unmodeled dynamics due to, e.g., congested routers.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Communication networks have evolved from specialized, research- and military-oriented transmission systems to large-scale and highly complex interconnections of intelligent devices. Networks today are heterogeneous and comprise not only of computer terminals, but also smart power grid networks [106, 115], ad hoc networks [122], sensor networks [2], and cognitive radio networks [70, 110]. The traffic carried by these networks has also increased explosively, thanks to the advances in processing speed and storage efficiency of electronic devices. The resulting need for higher efficiency in network operation and management has prompted researchers to rethink the principles of network design altogether. The last decade has thus witnessed significant advances in network information theory [48] and network optimization [150], as well as an increasing use of statistics and machine learning tools in communication networks [84].

From a systems perspective, all networks are distributed systems consisting of users that generate and consume traffic, along with protocols that regulate its flow across the network [150]. Effective operation of a network hinges on real-time allocation of network resources matching the user demands, while ensuring a prescribed minimum quality of service (QoS). Consequently, network operators are interested in (a) monitoring the network state in order to estimate the demand and availability of resources; and (b) designing protocols that allocate network resources to users in a fair and efficient manner. This thesis touches upon some of the key challenges encountered in both monitoring as well as resource allocation problems. Leveraging recent advances in network information and optimization theory, the first part of the thesis considers the problem of optimal resource allocation under different scenarios for both ad hoc and sensor networks. The second part of the thesis develops a statistical framework for network monitoring and prediction, with emphasis on large-scale networks and real-time operation.

1.1 Motivation and Context

The ability to deploy *ad hoc wireless networks* without centralized control or infrastructure is key to achieving the next-generation promise of ubiquitous connectivity. These self-organizing networks are already indispensable for applications such as sensing, combat support, search-and-rescue, as well as mesh networks. The challenge of efficiently utilizing the available resources, namely spectrum and battery life, is the first priority for ad hoc networks. The broad aim of the first part of this thesis is to systematically design and analyze resource-efficient ad hoc network protocols.

Internet protocols have been phenomenally successful in achieving high rates and ensuring low delays in wired networks. Indeed, most wireless routing schemes have sought to replicate the "wireline success" by neglecting the vagaries of the wireless interface, and reusing the available algorithms and protocols. However, this wireline mindset has generally led to inefficient use of bandwidth and power resources, as well as to considerable complications in the deployment of ad hoc networks. Network coding is a recent ground-breaking alternative to routing that offers the potential to transcend these arbitrary limits by embracing the peculiarities of wireless networks [94].

Network coding refers to the notion of allowing nodes to perform encoding operations on packets traversing the network [1]. Interestingly, linear mixing of packets is sufficient for achieving multicast capacity in wireline networks [61, 175]. This optimality result has encouraged harvesting the benefits of linear network coding to areas as diverse as distributed storage [44, 45], peer-assisted file delivery [66], streaming media [34, 118, 149], network tomography [58, 59], security [21, 77], data collection in sensor networks [164], and ad hoc networks [60]. The potential of network coding in wireless applications is also well appreci-

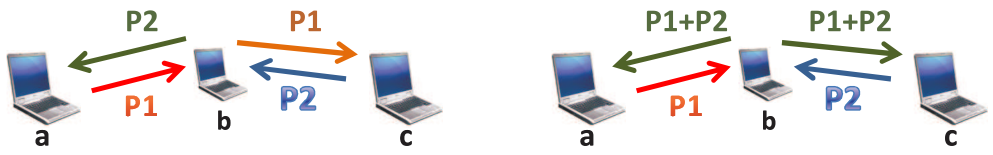


Figure 1.1: Information exchange through a relay node. Colors index the different time slots.

ated by now [28, 62], and some of the early prototypes include COPE [79], and MIXIT [80]. The emerging consensus is that network coding is not just an "exotic" routing substitute, but a new paradigm for information collection, storage, and dissemination.

To demonstrate the advantages of network coding over routing in wireless settings, Figure 1.1 depicts a canonical example, where two nodes a and c intend to exchange packets through an intermediate node b. With traditional routing (depicted on the left), this exchange requires at least four time slots. However, network coding at node b can allow the exchange to occur in just three time slots (as shown on the right). Observe how, unlike the routing scheme, network coding is also able to capitalize on the broadcast advantage inherent to wireless networks. This example underlines the importance of jointly designing network coding with transmission scheduling, in order to fully utilize the network capacity.

Switching our focus from optimization to monitoring, measurement tools are essential for maintaining seamless end-user experience in dynamic environments, as well as for ensuring network stability and security. In IP networks for instance, path delays and loss rates can portray network health, assess user experience, and allow users to compare different service providers. Unfortunately, acquisition and tracking of path metrics does not scale well to large networks, where the number of paths grows as the square of the number of end-points. This problem of "missing data" in Internet measurements has prompted the development of *inferential monitoring*, where statistical tools are used to impute the missing entries [84].

Early work in this context included network tomography, which aimed at inferring link delays in networks using only path delay measurements [24, 151]. Let the delay incurred on a link ℓ be denoted by x_ℓ, and let the $L \times 1$ vector $\mathbf{x}$ collect the delays on all links in a network. Also, let y_p be the path delay, collected in the $P \times 1$ vector $\mathbf{y}$ for all paths. The delay tomography approach utilizes the linear model

$$\mathbf{y} = \mathbf{G}\mathbf{x} \tag{1.1}$$

where the (p, ℓ)-th entry of the routing matrix $\mathbf{G}$ is one if path p includes link ℓ, and zero otherwise. Suppose further that at any time, only an $S \times 1$ sub-vector $\mathbf{y}^o$ with $S < L$ entries can be observed. Define the sub-vector of missing entries as $\mathbf{y}^u$, and the sub-matrices $\mathbf{G}_o$ and $\mathbf{G}_u$ formed by the rows corresponding to observed and unobserved paths, respectively. Then, the following expression was proposed in [24] for determining missing measurements

$$\hat{\mathbf{y}}^u = \mathbf{G}_u \mathbf{G}_o^T (\mathbf{G}_o \mathbf{G}_o^T)^{-1} \mathbf{y}^o \tag{1.2}$$

where the rows in $\mathbf{G}_o$ were chosen such $S = \text{rank}(\mathbf{G}) = \text{rank}(\mathbf{G}_o)$. Since $S \leq L \ll P$, this approach allows reduced measurement overhead compared to explicit measurement on all paths.

While network tomography allows perfect recovery of missing delays via (1.2), it cannot work if S is even one less than rank$(\mathbf{G})$. This imposes a severe limitation on its practicality, since measurement probes are always considered low-priority, and may easily get lost due to congestion. These considerations motivate monitoring via approximate techniques such as those employed in the context of spatial prediction [141], and these form the basis of the present work.

1.2 Thesis Outline and Contributions

The first part of this thesis proposes network coding protocols for ad hoc and sensor networks. In order to systematically design wireless protocols that can harness the full potential of network coding, a cross-layer design approach is pursued. Within this framework, protocols at different layers are allowed to interact with each other, in hopes of obtaining an improvement in network throughput and QoS. Subsections 1.2.1-1.2.4 describe the different scenarios under which protocol design is considered.

The second part of this thesis proposes a dynamic network delay cartography framework, which is described in Subsection 1.2.5.

1.2.1 Multicast in Fading Channels

As seen in Figure 1.1, even simple linear mixing operations can be powerful enough to enhance the network throughput, minimize delay, and decrease the overall power consumption [175], [28]. For the special case of single-source multicast, which does not even admit a polynomial-time solution within the routing framework [14], linear network coding achieves the full network capacity [1]. In fact, the network flow description of multicast with random network coding adheres to only linear inequality constraints reminiscent of the corresponding description in unicast routing [97].

This encourages the use of network coding to extend several popular results in unicast routing framework to multicast without appreciable increase in complexity. Of particular interest is the resource allocation and cross-layer optimization task in wireless networks [93], [65]. The objective here is to maximize a network utility function subject to flow, rate, capacity and power constraints. This popular approach not only offers the flexibility of capturing diverse performance objectives, but also admits a layering interpretation, arising from different decompositions of the optimization problem [26].

Chapter 2 deals with cross-layer optimization of wireless *multicast* networks that use network coding and operate over *fading* links. The aim is to maximize a total network utility objective, and entails finding end-to-end rates, network code design variables, broadcast link flows, link capacities, average power consumption, and instantaneous power allocations.

Network utility maximization was first brought into coded networks in [97], where the aim was to minimize a generic cost function subject only to flow and rate constraints. The optimal flow and rate variables may then be converted to a practical random network coding implementation using methods from [95] and [27]. Subsequent works extended this framework to include power, capacity, and scheduling constraints [3,38,159,170]. The interaction of network coding with the network and transport layers has also been explored in [23, 89, 91, 168, 169]; in these works, networks with fixed link capacities are studied, and different decomposition techniques result in different types of layered architectures.

There are however caveats associated with the utility maximization problem in wireless networks. First, the power control and scheduling subproblems are usually non-convex. This implies that the dual decomposition of the overall problem, though insightful, is not necessarily optimal and does not directly result in a feasible primal solution. Second, for continuous fading channels, determining the power control policy is an infinite dimensional problem. Existing approaches in network coding consider either deterministic channels [3, 170], or, links with a finite number of fading states [38, 74, 173].

On the other hand, a recent result in unicast routing shows that albeit the non-convexity, the overall utility optimization problem has no duality gap for wireless networks with continuous fading channels [139]. As this is indeed the case in all real-life fading environments, the result promises the optimality of layer separation. In particular, it renders a dual subgradient descent algorithm for network design optimal [64].

Chapter 2 begins with a formulation that jointly optimizes end-to-end rates, virtual flows, broadcast link flows, link capacities, average power consumption, and instantaneous power allocations in wireless fading multicast networks that use intra-session network coding. The first contribution of this chapter is to introduce a *realistic physical layer model* formulation accounting for the capacity of broadcast links. The cross-layer problem is generally non-convex, yet it is shown to have zero duality gap. This result considerably broadens [139] to *coded multicast* networks with broadcast links. The zero duality gap is then leveraged in order to develop a subgradient descent algorithm that minimizes the dual function. The algorithm admits a natural layering interpretation, allowing optimal integration of network coding into the protocol stack.

Next, the subgradient algorithm is modified so that the component of the subgradient that results from the physical layer power allocation may be delayed with respect to operations in other layers. This provably convergent asynchronous subgradient method and its *online* implementation constitute the second major contribution. Unlike the algorithm in [64], which is used for offline network optimization, the algorithm developed here is suitable for online network control. Convergence of asynchronous subgradient methods for dual minimization is known under diminishing stepsize [83]; this chapter proves results for constant stepsize. Near-optimal primal variables are also recovered by forming running averages of the primal iterates. This technique has also been used in synchronous subgradient methods for convex

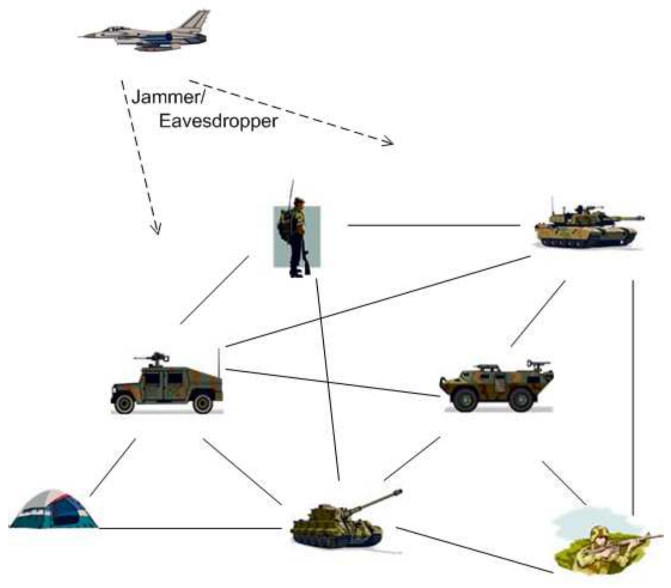


Figure 1.2: A tactical network

optimization; see e.g., [113] and references therein. Here, ergodic convergence results are established for the asynchronous scheme and the non-convex problem at hand.

1.2.2 Multicast in Random Access Networks

Tactical wireless ad hoc networks play a crucial role when it comes to communication dominance in the battlefield. Important requirements for such networks include resilience and efficiency. In order to accommodate units such as soldiers, military vehicles, or a field hospital, tactical networks are typically multi-hop; see e.g., Figure 1.2. Thus, it becomes important to deploy decentralized protocols, so that no single node exposes vulnerability of the network. Aloha is a simple, yet widely deployed medium access control (MAC) protocol, whose operation is distributed and resilient to both random, and jamming-induced link failures.

Chapter 3 focuses on multicasting applications for tactical networks, where information needs to be multicast from a single source to multiple target nodes. Efficient multicasting is realized using network coding whereby nodes perform encoding functions on packets traveling in the network. Although the multicast capacity region of wireless networks is not known, the rate region achievable with linear network coding has been characterized [97, 159]. This rate region can be practically achieved by fully distributed random linear network coding strategies [27, 72, 96]. Random network coding also results in each packet getting distributed spatially, thus providing some inherent protection against eavesdropping. Moreover, wedding of network coding with Aloha is particularly attractive for military networks because the network operation becomes extremely simple. Specifically, given the access probabilities, each node simply transmits random linear combinations of the packets in its buffer at a pre-specified rate [27, 72, 96]. The protocol does not require ACKs (nor retransmissions) at the MAC or network layers.

This chapter considers joint design of wireless *multi-hop* networks employing random network coding and slotted Aloha. A cross-layer optimization problem is formulated, where network coding rates (also called subgraphs) and transmission probabilities are jointly determined to maximize a network-wide objective. In contrast to simple protocol operation, the joint design itself is notoriously difficult. This is because the Aloha capacity region, even for three nodes with fully backlogged queues is described by non-convex signomial constraints [138].

Joint design of network coding and Aloha MAC has been undertaken for *single-hop* network topologies. The performance of slotted Aloha for star networks was analyzed in [88]. A game theoretic approach for throughput maximization in a single-hop setting was proposed in [75]. The performance for two-hop (relay) networks with bi-directional traffic was reported in [161]. These works underline the significance of the cross-layer approach for coded Aloha networks.

Joint design of coded Aloha multi-hop networks has been attempted. A branch-and-bound method was employed in [140] to obtain globally optimal transmission probabilities and subgraphs. While offering a benchmark for comparison, the resultant protocol may be too complex for use in large networks. A heuristic algorithm was proposed in [160], where the access probabilities and network coding rates were optimized separately. Albeit practical, the approach in [160] is suboptimal, and does not provide performance guarantees.

In this chapter, a successive convex approximation approach is adopted to obtain solutions that are guaranteed to be locally optimal. Convex surrogate problems are constructed in such a way to guarantee convergence of the overall algorithm to a Karush-Kuhn-Tucker (KKT) point of the original non-convex problem, and thus enable a tractable, locally optimal solution, even for large networks. This requires an efficient re-formulation of the MAC constraints, which constitutes our first contribution.

The constructed surrogate problems are not amenable to distributed solution. To this end, a separable structure is created by further approximating the problem, while still preserving KKT optimality of the overall algorithm. This forms our second contribution, which involves approximating even convex terms in order to make the overall problem separable. The dual subgradient method is employed for the resulting convex problems, where the primal and dual updates can be performed in a parallel and distributed fashion. A primal solution yielding near-optimal access probabilities and network coding rates, is recovered by primal averaging. An online network control protocol is also introduced to perform the optimization task.

The coded Aloha scheme enjoys features attractive for tactical networks. Specifically, the resulting protocol is simple and decentralized, whereby every node transmits random linear combinations with its access probability. Moreover, the optimal designs of this work take into account the packet loss probability due to the wireless medium (erasures); this feature can be leveraged to ensure jamming-resilience by preemptively setting higher erasure probabilities for any part of the network that is likely to be jammed. Furthermore, the subgradient-based online optimization and control uses a constant stepsize, which enables adaptation to slowly time-varying environments, for instance, due to mobility of branch units, or non-stationary jamming. The proposed scheme can also be used in low-end systems which do not implement any scheduling and power-allocation schemes.

1.2.3 Muticast under Delay Constraints

An important, but often overlooked, aspect of several wireless applications is the sensitivity of packets to delays. Streaming media and real-time sensor data, for example, are associated with strict deadlines, failing which, packets become useless. However, many wireless network coding implementations, such as [27], operate under the assumption of large block-lengths. This requires the sinks to accumulate a large number of packets before commencing the decoding process, thereby incurring prohibitively large delays.

Chapter 4 develops a joint scheduling and network coding (JS-NC) algorithm for wireless networks with packet delay constraints. A single source multicast scenario is considered where packets must be decoded at each sink within a specified number of time-slots since their first transmission by the source node. Delay constraints significantly complicate the JS-NC design since the optimal codes may have infinite block-lengths; see [50] and other references in Section 4.

Since infinite block-length codes are difficult to design as well as implement, a simpler periodic version of this joint design problem is proposed that operates on a time-unwrapped graph, thus allowing for finite block-length network codes. The *periodic formulation* is employed to derive a constant-factor approximate, augmenting-path design algorithm that is both scalable and distributed. The resultant network coding protocol does not require any end-to-end feedback or asymptotically large field size, and needs only a brief set-up time.

For networks with primary interference constraints, the JS-NC design problem is also analyzed from an integer programming perspective. A set of valid inequalities is developed which is subsequently used to derive a linear programming upper bound on the achievable throughput. Finally, simulations are used to corroborate the performance of the approximate JS-NC algorithm, and the quality of the associated bounds.

1.2.4 Network-Compressive Coding in Wireless Sensor Networks

Wireless sensor networks (WSNs) have become ubiquitous for cost-effective, distributed environment-monitoring and surveillance applications [156]. Deployed over large areas, WSNs are comprise of low-cost autonomous sensing devices with limited processing capabilities and battery life. In large-scale WSN deployments, however, relaying information over several hops becomes increasingly energy inefficient. On the other hand, observations from nearby sensors may be highly correlated; for instance, in temperature monitoring or intrusion-detection systems. For such applications, spatial correlation can be exploited to perform in-network compression of data, and achieve significant energy savings and prolonged network lifetime [119].

Chapter 5 develops network-compression algorithms that use *linear network coding* (LNC) to compress and communicate sensor observations. Compression via LNC features simple operations per sensor and reduced transmission energy. In general, the task of jointly designing data collection and compression protocols falls under the broad area of distributed source coding (DSC) [123, 146, 171]. However, in contrast to most DSC schemes, typically involving Slepian-Wolf coding, the LNC-based network-compression does not require the intermediate nodes to have knowledge of the correlation between sensor observations.

The use of network coding for in-network compression has been considered before in the context of network multicast; see [101] and references therein. Since optimal source-network decoding generally requires a search over an exponentially-large structured set of hypotheses, most results focus on characterizing the achievable rate region. As pointed out in [101], it is possible to perform approximate decoding by modeling the probabilistic relationships among observations using a factor graph [86], thereby allowing the use of low-complexity message-passing algorithms. The caveat though is that construction and analysis of "good" factor graphs, promising low decoding complexity, or reliably decoded symbol estimates, is not straightforward, and was not dealt with in [101].

This chapter considers the design and analysis of network-compressive coding and decoding algorithms. Using the sum-product algorithm for decoding, specific scenarios are identified, which yield factor graphs that admit practical protocols and low decoding error. Two novel factor graph constructions are proposed, offering complementary strengths in modeling and inference accuracy. Performance of the proposed approach is also analyzed by deriving error exponents of the probability that the distortion at the sink surpasses a given tolerable level. These error exponents expose the interplay between correlation level, compression ratio and alphabet size. The proposed algorithm is tested both on synthetic as well as real data sets, thus verifying its efficacy.

It is worth noting that the problem of efficiently collecting distributed data has also been explored in the context of decentralized detection, see e.g., [156] and references therein. However, most of these approaches are for scalar random variables [145], and assume that all sensors receive observations from the same variable [180]. Some approaches assume the observations to be real-valued and exploit compressive-sensing [69], or Gaussian belief propagation [7] for recovery. However, these algorithms entail mixing and transmission of analog-amplitude messages, which may be impractical in low-cost sensing devices. Moreover, none of the existing approaches considers the design of mixing matrices (tailored to minimize communication cost), or analyze the impact of quantization errors.

1.2.5 Dynamic Network Cartography

The explosive growth in network size has necessitated the development of avant-garde monitoring tools to endow network operators with a real-time view of the global network behavior. As pointed out earlier, acquisition and processing of network-wide performance metrics for large networks is no easy task. Focus has thus shifted towards statistical means of predicting network-wide performance metrics using measurements on only a subset of nodes [124, 153]. A promising approach in this context has been the application of *kriging*, a tool for spatial prediction popular in geostatistics and environmental sciences [36, 141]. A *network kriging* approach was developed in [29], where network-wide path delays were predicted using measurements on a chosen subset of paths. The class of linear predictors introduced leverages network topology information to model the covariance among path delays. This is accomplished in [29] by assigning higher correlation between two paths if they share several links, as in this case, they are expected to incur similar delay variations.

Chapter 6 puts forth a *dynamic* network kriging approach capable of real-time spatio-temporal delay predictions. Specifically, a kriged Kalman filter (KKF) is employed to explicitly capture variations due to queuing delays, while retaining the topology-based kriging predictor. The resulting dynamic network kriging approach not only yields lower prediction error, but is also more flexible, allowing delay measurements to be taken on random subsets of paths. In this context, the problem of choosing the optimal paths for delay measurements is also considered. Since the KKF runs in real-time, the paths are also selected in an online fashion by minimizing the prediction error per time slot. Interestingly, the resulting combinatorial optimization problem is shown to be submodular, and is therefore solved approximately via a greedy routine.

Recently, a compressive sampling-based approach has also been reported for predicting network-wide performance metrics [30, 172]. For instance, diffusion wavelets were utilized in [30] to obtain a compressible representation of the delays, and account for spatial and temporal correlations. Although this allows for enhanced prediction accuracy over [29], it requires batch processing of measurements which does not scale well to large networks for real-time operation. In contrast, both the KKF and the greedy path selection algorithms entail sequential operations, and are therefore significantly faster.

Imputation of end-to-end delays has also been considered in the context of Internet geolocation. Treating end-to-end delays as distances between nodes, all-pair node distances are estimated using Euclidean embedding [40], or, matrix factorization [92]. However, these approaches do not exploit the temporal or topological information, since their focus is not on monitoring or extrapolation (that is, prediction) of delays.

1.3 Publications

The present Ph.D. work on network optimization and monitoring has resulted in publication of three journal papers (in the IEEE/ACM Transactions on Networking [130], IEEE Transactions on Signal Processing [134], and IEEE Journal on Selected Areas in Communications [131]). It has also led to two journal submissions, currently under consideration for publication (in the IEEE Transactions of Wireless Communications [126] and IEEE Transactions of Information Theory [129]), and one journal paper in preparation [56] for submission to the Journal of Machine Learning Research. In addition to these 6 journal papers, results in this thesis have also been disseminated at pertinent conferences, where a total of 7 conference articles has been accepted for publication [57,127,128,132,133,135,136].

Chapter 2

Cross-Layer Design of Coded Multicast in Fading

This chapter deals with cross-layer designs in wireless fading networks. An optimal resource allocation framework is formulated, where the nodes are allowed to perform network coding. The aim is to jointly optimize end-to-end transport layer rates, network code design variables, broadcast link flows, link capacities, average power consumption, and short-term power allocation policies. As in the routing paradigm where nodes simply forward packets, the cross-layer optimization problem with network coding is non-convex in general. It is proved however, that with network coding, dual decomposition for multicast is optimal so long as the fading at each wireless link is a continuous random variable. This lends itself to provably convergent subgradient algorithms, which not only admit a layered-architecture interpretation but also optimally integrate network coding in the protocol stack. The dual algorithm is also paired with a scheme that yields near-optimal network design variables, namely multicast end-to-end rates, network code design quantities, flows over the broadcast links, link capacities, and average power consumption. Finally, an asynchronous subgradient method is developed, whereby the dual updates at the physical layer can be affordably performed with a certain delay with respect to the resource allocation tasks in upper layers. This attractive feature is motivated by the complexity of the physical layer subproblem, and is an adaptation of the subgradient method suitable for network control.

The organization of this chapter is as follows. Section 2.1 presents the problem formulation that jointly optimizes end-to-end rates, virtual flows, broadcast link flows, link capacities, average power consumption, and instantaneous power allocations in wireless fading multicast networks that use intra-session network coding. The cross-layer problem is generally non-convex, yet it is shown to have zero duality gap (Section 2.2.1). The zero duality gap is then leveraged in order to develop a subgradient descent algorithm that minimizes the dual function (Section 2.2.2), and is provably convergent (Section 2.2.3). In Section 2.3, the subgradient algorithm is modified so that the component of the subgradient that results from the physical layer power allocation may be delayed with respect to operations in other layers. Finally, numerical results are presented in Section 2.4, and Section 2.5 concludes the chapter.

2.1 Problem Formulation

Consider a wireless network consisting of a set of terminals (nodes) denoted by $\mathcal{N}$. The broadcast property of the wireless interface is modeled by using the concept of hyperarcs. A *hyperarc* is a pair (i, J) that represents a broadcast link from a node i to a chosen set of nodes $J \subset \mathcal{N}$. The entire network can therefore be represented as a hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{A}$ is the set of hyperarcs. The complexity of the model is determined by the choice of the set $\mathcal{A}$. Let the neighbor-set $N(i)$ denote the set of nodes that node i reaches. An exhaustive model might include all possible $2^{|N(i)|} - 1$ hyperarcs from node i. On the other hand, a simpler model might include only a smaller number of hyperarcs per node. A point-to-point model is also a special case when node i has $|N(i)|$ hyperarcs each containing just one receiver.

The present chapter considers a physical layer whereby the channels undergo random multipath fading. This model allows for opportunistically best schedules per channel realization. This is different from the link-level network models in [38, 74, 97, 159], where the hyperarcs are modeled as erasure channels. The next subsection discusses the physical layer model in detail.

2.1.1 Physical Layer

In the current setting, terminals are assumed to have a set of tones $\mathcal{F}$ available for transmission. Let h_{ij}^f denote the power gain between nodes i and j over a tone $f \in \mathcal{F}$, assumed random, capturing fading effects. Let $\mathbf{h}$ represent the vector formed by stacking all the channel gains. The network operates in a time slotted fashion; the channel $\mathbf{h}$ remains constant for the duration of a slot, but is allowed to change from slot to slot. A slowly fading channel is assumed so that a large number of packets may be transmitted per time slot. The fading process is modeled to be stationary and ergodic.

Since the channel changes randomly per time slot, the optimization variables at the physical layer are the channel realization-specific power allocations $p_{ij}^f(\mathbf{h})$ for all hyperarcs $(i, J) \in \mathcal{A}$, and tones $f \in \mathcal{F}$. For convenience, these power allocations are stacked in a vector $\mathbf{p}(\mathbf{h})$. Instantaneous power allocations may adhere to several scheduling and mask constraints, and these will be generically denoted by a bounded set Π such that $\mathbf{p}(\mathbf{h}) \in \Pi$. The long-term average power consumption by a node i is given by

$$p_i = \mathbb{E}\left[\sum_f \sum_{J:(i,J)\in\mathcal{A}} p_{iJ}^f(\mathbf{h})\right] \tag{2.1}$$

where $\mathbb{E}[.]$ denotes expectation over the stationary channel distribution.

For slow fading channels, the information-theoretic capacity of a hyperarc (i, J) is defined as the maximum rate at which *all* nodes in J receive data from i with vanishing probability of error in a given time slot. This capacity depends on the instantaneous power allocations $\mathbf{p}(\mathbf{h})$ and channels $\mathbf{h}$. A generic bounded function $C_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h})$ will be used to describe this mapping. Next we give two examples of the functional forms of $C_{iJ}^f(\cdot)$ and Π.

Example 2.1. Conflict graph model: The power allocations p_{ij}^f adhere to the spectral mask constraints

$$0 \le p_{ij}^f \le p_{\max}^f. \tag{2.2}$$

However, only conflict-free hyperarc are allowed to be scheduled for a given $\mathbf{h}$. Specifically, power may be allocated to hyperarcs (i_1, J_1) and (i_2, J_2) if and only if [159]

i) $i_1 \neq i_2$;

ii) $i_1 \notin J_2$ and $i_2 \notin J_1$ (half-duplex operation); and

iii-a) $J_1 \cap J_2 = \emptyset$ (primary interference), or additionally,

iii-b) $J_1 \cap N(i_2) = J_2 \cap N(i_1) = \emptyset$ (secondary interference).

The set Π therefore consists of all possible power allocations that satisfy the previous properties.

Due to hyperarc scheduling, all transmissions in the network are interference free. The signal-to-noise ratio (SNR) at a node $j \in J$ is given by

$$\Gamma_{iJj}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}) = \frac{p_{ij}^f(\mathbf{h}) h_{ij}^f}{N_j} \tag{2.3}$$

where N_j is the noise power at j. In a broadcast setting, the maximum rate of information transfer from i to *each* node in J is

$$C_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}) = \min_{j \in J} \log(1 + \Gamma_{iJj}^f(\mathbf{p}(\mathbf{h}), \mathbf{h})). \tag{2.4}$$

A similar expression can be written for the special case of point-to-point links by substituting hyperarcs (i, J) by arcs (i, j) in the expression for $\Gamma_{iJj}^f(\mathbf{p}(\mathbf{h}), \mathbf{h})$.

For slow-fading channels, Gaussian codebooks with sufficiently large block lengths achieve this capacity in every time slot. More realistically, an SNR penalty term ρ can be included to account for finite-length practical codes and adaptive modulation schemes, so that

$$C_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}) = \min_{j \in J} \log\left(1 + \Gamma_{iJj}^f(\mathbf{p}(\mathbf{h}), \mathbf{h})/\rho\right). \tag{2.5}$$

The penalty term is in general a function of the target bit error rate.

Example 2.2. Signal-to-interference-plus-noise-ratio (SINR) model: Here, the constraint set Π is simply a box set $\mathcal{B}_\mathbf{p}$,

$$\Pi = \mathcal{B}_\mathbf{p} := \{p_{ij}^f | 0 \le p_{ij}^f \le p_{\max}^f \; \forall \, (i, J) \in \mathcal{A} \text{ and } f \in \mathcal{F} \}. \tag{2.6}$$

The set $\mathcal{B}_\mathbf{p}$ could also include (instantaneous) sum-power constraints per node. The capacity is expressed as in (2.4) or (2.5), but now the SNR is replaced by the SINR, given by

$$\Gamma_{iJj}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}) = p_{ij}^f(\mathbf{h}) h_{ij}^f \Big/ \left(N_j + I_{ij,f}^{\text{int}} + I_{j,f}^{\text{self}} + I_{iJj,f}^{\text{broad}}\right). \tag{2.7}$$

The denominator consists of the following terms:

- Interference from other nodes' transmissions to node j

$$I_{ij,f}^{\text{int}} = \sum_{\substack{(k,M)\in\mathcal{A}:j\in M,\\ k\neq j, k\neq i}} p_{kM}^{f}(\mathbf{h}) h_{kj}^{f}. \tag{2.8a}$$

- "Self-interference" due to transmissions of node j

$$I_{j,f}^{\text{self}} = h_{jj} \sum_{M:(j,M)\in\mathcal{A}} p_{jM}^{f}(\mathbf{h}). \tag{2.8b}$$

  This term is introduced to encourage half-duplex operation by setting h_{jj} to a large value.

- "Broadcast-interference" from transmissions of node i to other hyperarcs

$$I_{iJj,f}^{\text{broad}} = \beta h_{ij}^{f} \sum_{\substack{M:(i,M)\in\mathcal{A}\\ M\neq J}} p_{iM}^{f}(\mathbf{h}). \tag{2.8c}$$

  This term is introduced to force node i to transmit at most over a single hyperarc, by setting β to a large value.

The previous definitions ignore interference from non-neighboring nodes. However, they can be readily extended to include more general interference models.

The link layer capacity is defined as the long-term average of the total instantaneous capacity, namely,

$$c_{iJ} := \mathbb{E}\left[\sum_{f} C_{iJ}^{f}(\mathbf{p}(\mathbf{h}), \mathbf{h})\right]. \tag{2.9}$$

This is also called ergodic capacity and represents the maximum average data rate available to the link layer.

2.1.2 Link Layer and Above

The network supports multiple multicast sessions indexed by m, namely $\mathcal{S}_m := (s^m, T^m, a^m)$, each associated with a source node s^m, sink nodes $T^m \subset \mathcal{N}$, and an average flow rate a^m from s^m to each $t \in T^m$. The value a^m is the average rate at which the network layer of source terminal s^m admits packets from the transport layer. Traffic is considered elastic, so that the packets do not have any short-term delay constraints.

Network coding is a generalization of routing since the nodes are allowed to code packets together rather than simply forward them. This chapter considers intra-session network coding, where only the traffic belonging to the same multicast session is allowed to mix. Although better than routing in general, this approach is still suboptimal in terms of achieving the network capacity. However, general (inter-session) network coding is difficult to characterize or implement since neither the capacity region nor efficient network code designs are known [175, Part II]. On the other hand, a simple linear coding strategy achieves the full capacity region of intra-session network coding [1].

The network layer consists of endogenous flows of coded packets over hyperarcs. Recall that the maximum average rate of transmission over a single hyperarc cannot exceed c_{iJ}. Let the coded packet-rate of a multicast session m over hyperarc (i, J) be z_{iJ}^m (also referred to as the subgraph or broadcast link flow). The link capacity constraints thus translate to

$$\sum_m z_{iJ}^m \leq c_{iJ} \quad \forall (i, J) \in \mathcal{A}. \tag{2.10}$$

To describe the intra-session network coding capacity region, it is commonplace to use the concept of *virtual flow* between terminals i and j corresponding to each session m and sink $t \in T^m$ with average rate x_{ij}^{mt}. These virtual flows are defined only for neighboring pairs of nodes i.e., $(i, j) \in \mathcal{G} := \{(i, j) | (i, J) \in \mathcal{A}, j \in J\}$. The virtual flows satisfy the flow-conservation constraints, namely,

$$\sum_{j:(i,j)\in\mathcal{G}} x_{ij}^{mt} - \sum_{j:(j,i)\in\mathcal{G}} x_{ji}^{mt} = \sigma_i^m := \begin{cases} a^m & \text{if } i = s^m, \\ -a^m & \text{if } i = t, \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

for all m, $t \in T^m$, and $i \in \mathcal{N}$. Hereafter, the set of equations for $i = t$ will be omitted because they are implied by the remaining equations.

The broadcast flows z_{iJ}^m and the virtual flows x_{ij}^{mt} can be related using results from the lossy-hyperarc model of [97, 159]. Specifically, [159, eq. (9)] relates the virtual flows and subgraphs, using the fraction $b_{iJK} \in [0, 1]$ of packets injected into the hyperarc (i, J) that reach the set of nodes $K \subset N(i)$. Recall from Section 2.1.1, that here the instantaneous capacity function $C_{iJ}^{f}(\cdot)$ is defined such that all packets injected into the hyperarc (i, J) are received by every node in J. Thus in our case, $b_{iJK} = 1$ whenever $K \cap J \neq \emptyset$ and consequently,

$$\sum_{j \in K} x_{ij}^{mt} \leq \sum_{\substack{J:(i,J)\in\mathcal{A} \\ J\cap K\neq\emptyset}} z_{iJ}^{m}, \quad K \subset N(i), i \in \mathcal{N}, m, t \in T^m. \tag{2.12}$$

Note the difference with [159] where at every time slot, packets are injected into a fixed set of hyperarcs at the same rate. The problem in [159] is therefore to find a schedule of hyperarcs that do not interfere (the non-conflicting hyperarcs). The same schedule is used at every time slot; however, only a random subset of nodes receive the injected packets in a given slot. Instead here, the hyperarc selection is part of the power allocation problem at the physical layer, and is done for every time slot. The transmission rate (or equivalently, the channel coding redundancy) is however appropriately adjusted so that all the nodes in the selected hyperarc receive the data.

In general, for any feasible solution to the set of equations (2.10)–(2.12), a network code exists that supports the corresponding exogenous rates a^m [97]. This is because for each multicast session m, the maximum flow between s^m and $t \in T^m$ is a^m, and is therefore achievable [1, Th. 1]. Given a feasible solution, various network coding schemes can be used to achieve the exogenous rates. Random network coding based implementations such as those proposed in [95] and [27], are particularly attractive since they are fully distributed and require little overhead. These schemes also handle any residual errors or erasures that remain due to the physical layer.

The system model also allows for a set of "box constraints" that limit the long-term powers, transport layer rates, broadcast link flow rates, virtual flow rates as well as the maximum link capacities. Combined with the set Π, these constraints can be compactly expressed as

$$\mathcal{B} := \{\mathbf{y}, \mathbf{p}(\mathbf{h}) | \ \mathbf{p}(\mathbf{h}) \in \Pi, \ \ 0 \leq p_i \leq p_i^{\max}, \ \ a_{\min}^m \leq a^m \leq a_{\max}^m, \ \ 0 \leq c_{iJ} \leq c_{iJ}^{\max},$$
$$0 \leq z_{iJ}^m \leq z_{iJ}^{\max}, \ \ 0 \leq x_{ij}^{mt} \leq x_{ij}^{\max}\}. \tag{2.13}$$

Here $\mathbf{y}$ is a super-vector formed by stacking all the average rate and power variables, that is, a^m, z_{iJ}^m, x_{ij}^{mt}, c_{iJ}, and p_i. Parameters with min/max subscripts or superscripts denote prescribed lower/upper bounds on the corresponding variables.

2.1.3 Optimal Resource Allocation

A common objective of the network optimization problem is maximization of the exogenous rates a^m and minimization of the power consumption p_i. Towards this end, consider increasing and concave utility functions $U_m(a^m)$ and convex cost functions $V_i(p_i)$ so that the overall objective function $f(\mathbf{y}) = \sum_m U_m(a^m) - \sum_i V(p_i)$ is concave. For example, the utility function can be the logarithm of session rates and the cost function can be the squared average power consumption. The network utility maximization problem can be written as

$$\mathsf{P} = \max_{(\mathbf{y},\mathbf{p}(\mathbf{h}))\in\mathcal{B}} \sum_m U_m(a^m) - \sum_i V_i(p_i) \tag{2.14a}$$

$$\text{s. t. } \sigma_i^m \le \sum_{(i,j)\in\mathcal{G}} x_{ij}^{mt} - \sum_{(j,i)\in\mathcal{G}} x_{ji}^{mt} \qquad \forall\, m, i \neq t, t \in T^m \tag{2.14b}$$

$$\sum_{j\in K} x_{ij}^{mt} \le \sum_{\substack{J:(i,J)\in\mathcal{A}\\ J\cap K\neq\emptyset}} z_{iJ}^m \qquad \forall\, K \subset N(i), m, t \in T^m \tag{2.14c}$$

$$\sum_m z_{iJ}^m \le c_{iJ} \qquad \forall\, (i,J) \in \mathcal{A} \tag{2.14d}$$

$$c_{iJ} \le \mathbb{E}\left[\sum_f C_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h})\right] \qquad \forall\, (i,J) \in \mathcal{A} \tag{2.14e}$$

$$\mathbb{E}\left[\sum_f \sum_{J:(i,J)\in\mathcal{A}} p_{iJ}^f(\mathbf{h})\right] \le p_i \tag{2.14f}$$

where $i \in \mathcal{N}$. Note that constraints (2.1), (2.9) and (2.11) have been relaxed without increasing the objective function. For instance, the relaxation of (2.11) is equivalent to allowing each node to send at a higher rate than received, which amounts to adding virtual sources at all nodes $i \neq t$. However, adding virtual sources does not result in an increase in the objective function because the utilities U_m depend only on the multicast rate a^m.

The solution of the optimization problem (2.14) gives the throughput a^m that is achievable using optimal virtual flow rates x_{ij}^{mt} and power allocation policies $\mathbf{p}(\mathbf{h})$. These virtual flow rates are used for network code design. When implementing coded networks in practice, the traffic is generated in packets and stored at nodes in queues (and virtual queues for virtual flows) [27]. The constraints in (2.14) guarantee that all queues are stable.

Optimization problem (2.14) is non-convex in general, and thus difficult to solve. For example, in the conflict graph model, the constraint set Π is discrete and non-convex, while in the SINR-model, the capacity function $C_{iJ}^{f}(\mathbf{p}(\mathbf{h}), \mathbf{h})$ is a non-concave function of $\mathbf{p}(\mathbf{h})$; see e.g., [98], [93]. The next section analyzes the Lagrangian dual of (2.14).

2.2 Optimality of Layering

This section shows that (2.14) has zero duality gap, and solves the dual problem via subgradient descent iterations. The purpose here is two-fold: i) to describe a layered architecture in which linear network coding is optimally integrated; and ii) to set the basis for a network implementation of the subgradient method, which will be developed in Section 2.3.

2.2.1 Duality Properties

Associate Lagrange multipliers ν_i^{mt}, η_{iK}^{mt}, ξ_{iJ}, λ_{iJ} and μ_i with the flow constraints (2.14b), the union of flow constraints (2.14c), the link rate constraints (2.14d), the capacity constraints (2.14e), and the power constraints (2.14f), respectively. Also, let $\boldsymbol{\zeta}$ be the vector formed by stacking these Lagrange multipliers in the aforementioned order. Similarly, if inequalities (2.14b)–(2.14f) are rewritten with zeros on the right-hand side, the vector $\mathbf{q}(\mathbf{y}, \mathbf{p}(\mathbf{h}))$ collects all the terms on the left-hand side of the constraints. The Lagrangian can therefore be written as

$$\mathcal{L}(\mathbf{y}, \mathbf{p}(\mathbf{h}), \boldsymbol{\zeta}) = \sum_{m} U_m(a^m) - \sum_{i \in \mathcal{N}} V_i(p_i) - \boldsymbol{\zeta}^T \mathbf{q}(\mathbf{y}, \mathbf{p}(\mathbf{h})). \tag{2.15}$$

The dual function and the dual problem are, respectively,

$$\varrho(\boldsymbol{\zeta}) := \max_{(\mathbf{y}, \mathbf{p}(\mathbf{h})) \in \mathcal{B}} \mathcal{L}(\mathbf{y}, \mathbf{p}(\mathbf{h}), \boldsymbol{\zeta}) \tag{2.16}$$

$$\mathsf{D} = \min_{\boldsymbol{\zeta} \geq \mathbf{0}} \varrho(\boldsymbol{\zeta}). \tag{2.17}$$

Since (2.14e) may be a non-convex constraint, the duality gap is in general, non-zero; i.e., $\mathsf{D} \geq \mathsf{P}$. Thus, solving (2.17) yields an upper bound on the optimal value P of (2.14). In the present formulation however, we have the following interesting result.

Proposition 2.1. *If the fading is continuous, then the duality gap is exactly zero, i.e.,*

$$\mathsf{P} = \mathsf{D}. \tag{2.18}$$

A generalized version of Proposition 2.1, including a formal definition of continuous fading, is provided in Appendix 2.A and connections to relevant results are made. The essential reason behind this strong duality is that the set of ergodic capacities resulting from all feasible power allocations is convex.

The requirement of continuous fading channels is not limiting since it holds for all practical fading models, such as Rayleigh, Rice, or Nakagami-m. Recall though that the dual problem is always convex. The subgradient method has traditionally been used to approximately solve (2.17), and also provide an intuitive layering interpretation of the network optimization problem [26]. The zero duality gap result is remarkable in the sense that it renders this layering optimal.

A corresponding result for unicast routing in uncoded networks has been proved in [139]. The fact that it holds for coded networks with broadcast links, allows optimal integration of the network coding operations in the wireless protocol stack. The next subsection deals with this subject.

2.2.2 Subgradient Algorithm and Layer Separability

The dual problem (2.17) can in general be solved using the subgradient iterations [12, Section 8.2] indexed by ℓ

$$(\mathbf{y}(\ell), \mathbf{p}(\mathbf{h};\ell)) \in \underset{(\mathbf{y},\mathbf{p}(\mathbf{h}))\in\mathcal{B}}{\arg\max}\ \mathcal{L}(\mathbf{y}, \mathbf{p}(\mathbf{h}), \boldsymbol{\zeta}(\ell)) \tag{2.19a}$$

$$\boldsymbol{\zeta}(\ell+1) = [\boldsymbol{\zeta}(\ell) + \epsilon\mathbf{q}(\mathbf{y}(\ell), \mathbf{p}(\mathbf{h};\ell))]^{+} \tag{2.19b}$$

where ϵ is a positive constant stepsize, and $[.]^{+}$ denotes projection onto the nonnegative orthant. The inclusion symbol ($\in$) allows for potentially multiple maxima. In (2.19b), $\mathbf{q}(\mathbf{y}(\ell), \mathbf{p}(\mathbf{h};\ell))$ is a subgradient of the dual function $\varrho(\boldsymbol{\zeta})$ in (2.16) at $\boldsymbol{\zeta}(\ell)$. Next, we discuss the operations in (2.19) in detail.

For the Lagrangian obtained from (2.15), the maximization in (2.19a) can be separated into the following subproblems

$$a_i^m(\ell) \in \operatorname*{arg\,max}_{a^m_{\min} \le a^m \le a^m_{\max}} \left[U_m(a^m) - \sum_{t \in T^m} \nu_{s_m}^{mt}(\ell) a^m \right] \tag{2.20a}$$

$$z_{iJ}^m(\ell) \in \operatorname*{arg\,max}_{0 \le z_{iJ}^m \le z_{iJ}^{\max}} \left[\sum_{\substack{K \subset N(i) \\ K \cap J \neq \emptyset}} \sum_{t \in T^m} \eta_{iK}^{mt}(\ell) - \xi_{iJ}(\ell) \right] z_{iJ}^m \tag{2.20b}$$

$$x_{ij}^{mt}(\ell) \in \operatorname*{arg\,max}_{0 \le x_{ij}^{mt} \le x_{ij}^{\max}} \left[\nu_i^{mt}(\ell) \mathbf{1}_{i \neq t} - \nu_j^{mt}(\ell) \mathbf{1}_{j \neq t} - \sum_{\substack{K \subset N(i) \\ j \in K}} \eta_{iK}^{mt}(\ell) \right] x_{ij}^{mt} \tag{2.20c}$$

$$c_{iJ}(\ell) \in \operatorname*{arg\,max}_{0 \le c_{iJ} \le c_{iJ}^{\max}} \left[\xi_{iJ}(\ell) - \lambda_{iJ}(\ell) \right] c_{iJ} \tag{2.20d}$$

$$p_i(\ell) \in \operatorname*{arg\,max}_{0 \le p_i \le p_i^{\max}} \left[\mu_i(\ell) p_i - V_i(p_i) \right] \tag{2.20e}$$

$$\mathbf{p}(\mathbf{h}; \ell) \in \operatorname*{arg\,max}_{\mathbf{p}(\mathbf{h}) \in \Pi} \sum_{f,(i,J) \in \mathcal{A}} \gamma_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}, \boldsymbol{\zeta}) \tag{2.20f}$$

where

$$\gamma_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}, \boldsymbol{\zeta}) := \lambda_{iJ} C_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}) - \mu_i p_{ij}^f(\mathbf{h}) \tag{2.20g}$$

and $\mathbf{1}_X$ is the indicator function, which equals one if the expression X is true, and zero otherwise.

The physical layer subproblem (2.20f) implies per-fading state separability. Specifically, instead of optimizing over the class of power control policies, (2.20f) allows solving for the optimal power allocation for each fading state; that is,

$$\begin{aligned} \mathsf{P}(\mathbf{p}(\mathbf{h})) &= \max_{\mathbf{p}(\mathbf{h}) \in \Pi} \mathbb{E} \left[\sum_{f,(i,J) \in \mathcal{A}} \gamma_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}, \boldsymbol{\zeta}) \right] \\ &= \mathbb{E} \left[\max_{\mathbf{p}(\mathbf{h}) \in \Pi} \sum_{f,(i,J) \in \mathcal{A}} \gamma_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h}, \boldsymbol{\zeta}) \right]. \end{aligned} \tag{2.21}$$

Note that problems (2.20a)–(2.20e) are convex and admit efficient solutions. The per-fading state power allocation subproblem (2.20f) however, may not necessarily be convex. For example, under the conflict graph model (cf. Example 1), the number of feasible power allocations may be exponential in the number of nodes. Finding an allocation that maximizes the objective function in (2.21) is equivalent to the NP-hard maximum weighted hyperarc matching problem [159]. Similarly, the capacity function and hence the objective function for the SINR model (cf. Example 2) is non-convex in general, and may be difficult to optimize.

This separable structure allows a useful layered interpretation of the problem. In particular, the transport layer sub-problem (2.20a) gives the optimal exogenous rates allowed into the network; the network flow sub-problem (2.20b) yields the endogenous flow rates of coded packets on the hyperarcs; and the virtual flow sub-problem (2.20c) is responsible for determining the virtual flow rates between nodes and therefore the network code design. Likewise, the capacity sub-problem (2.20d) yields the link capacities and the power sub-problem (2.20e) provides the power control at the data link layer.

The layered architecture described so far also allows for optimal integration of network coding into the protocol stack. Specifically, the broadcast and virtual flows optimized respectively in (2.20b) and (2.20c), allow performing the combined routing-plus-network coding task at network layer. An implementation such as the one in [27] typically requires queues for both broadcast as well as virtual flows to be maintained here.

Next, the subgradient updates of (2.19b) become

$$\nu_i^{mt}(\ell+1) = \left[\nu_i^{mt}(\ell) + \epsilon\check{q}_\nu^{imt}(\ell)\right]^+ \tag{2.22a}$$

$$\eta_{iK}^{mt}(\ell+1) = \left[\eta_{iK}^{mt}(\ell) + \epsilon\check{q}_\eta^{iKmt}(\ell)\right]^+ \tag{2.22b}$$

$$\xi_{iJ}(\ell+1) = \left[\xi_{iJ}(\ell) + \epsilon\check{q}_\xi^{iJ}(\ell)\right]^+ \tag{2.22c}$$

$$\lambda_{iJ}(\ell+1) = \left[\lambda_{iJ}(\ell) + \epsilon\check{q}_\lambda^{iJ}(\ell)\right]^+ \tag{2.22d}$$

$$\mu_i(\ell+1) = \left[\mu_i(\ell) + \epsilon\check{q}_\mu^{i}(\ell)\right]^+ \tag{2.22e}$$

where $\check{q}(\ell)$ are the subgradients at index ℓ given by

$$\check{q}_{\nu}^{imt}(\ell) = \sigma_i^m(\ell) + \sum_{(i,j)\in\mathcal{G}} x_{ji}^{mt}(\ell) - \sum_{(j,i)\in\mathcal{G}} x_{ij}^{mt}(\ell) \tag{2.23a}$$

$$\check{q}_{\eta}^{iKmt}(\ell) = \sum_{j\in K} x_{ij}^{mt}(\ell) - \sum_{\substack{J:(i,J)\in\mathcal{A}\\ J\cap K\neq\emptyset}} z_{iJ}^{m}(\ell) \tag{2.23b}$$

$$\check{q}_{\xi}^{iJ}(\ell) = \sum_{m} z_{iJ}^{m}(\ell) - c_{iJ}(\ell) \tag{2.23c}$$

$$\check{q}_{\lambda}^{iJ}(\ell) = c_{iJ}(\ell) - \mathbb{E}\left[\sum_f C_{iJ}^f(\mathbf{p}(\mathbf{h};\ell),\mathbf{h})\right] \tag{2.23d}$$

$$\check{q}_{\mu}^{i}(\ell) = \mathbb{E}\left[\sum_f \sum_{J:(i,J)\in\mathcal{A}} p_{iJ}^f(\mathbf{h};\ell)\right] - p_i(\ell). \tag{2.23e}$$

The physical layer updates (2.22d) and (2.22e) are again complicated since they involve the $\mathbb{E}[.]$ operations of (2.23d) and (2.23e). These expectations can be acquired via Monte Carlo simulations by solving (2.20f) for realizations of $\mathbf{h}$ and averaging over them. These realizations can be independently drawn from the distribution of $\mathbf{h}$, or they can be actual channel measurements. In fact, the latter is implemented in Section 2.3 on the fly during network operation.

2.2.3 Convergence Results

This subsection provides convergence results for the subgradient iterations (2.19). Since the primal variables $(\mathbf{y}, \mathbf{p}(\mathbf{h}))$ and the capacity function $C_{iJ}^f(.)$ are bounded, it is possible to define an upper bound G on the subgradient norm; i.e., $\|\mathbf{q}(\mathbf{y}(\ell), \mathbf{p}(\mathbf{h};\ell))\| \leq G$ for all $\ell \geq 1$.

Proposition 2.2. *For the subgradient iterations in* (2.20) *and* (2.22), *the best dual value converges to* D *upto a constant; i.e.,*

$$\lim_{s\to\infty} \min_{1\leq\ell\leq s} \varrho(\boldsymbol{\zeta}(\ell)) \leq \mathsf{D} + \frac{\epsilon G^2}{2}. \tag{2.24}$$

This result is well known for dual (hence, convex) problems [12, Prop. 8.2.3]. However, the presence of an infinite-dimensional variable $\mathbf{p}(\mathbf{h})$ is a subtlety here. A similar case is dealt with in [139] and Proposition 2.2 follows from the results there.

Note that in the subgradient method (2.19), the sequence of primal iterates $\{\mathbf{y}(\ell)\}$ does not necessarily converge. However, a primal running average scheme can be used for finding the optimal primal variables $\mathbf{y}^*$ as summarized next. Recall that $f(\mathbf{y})$ denotes the objective function $\sum_m U_m(a^m) - \sum_i V_i(p_i)$.

Proposition 2.3. *For the running average of primal iterates*

$$\bar{\mathbf{y}}(s) := \frac{1}{s}\sum_{\ell=1}^{s}\mathbf{y}(\ell). \tag{2.25}$$

the following results hold:

a) There exists a sequence $\{\mathring{\mathbf{p}}(\mathbf{h}; s)\}$ such that $(\bar{\mathbf{y}}(s), \mathring{\mathbf{p}}(\mathbf{h}; s)) \in \mathcal{B}$, and also

$$\lim_{s\to\infty} \left\| [\mathbf{q}(\bar{\mathbf{y}}(s), \mathring{\mathbf{p}}(\mathbf{h}; s))]^+ \right\| = 0. \tag{2.26}$$

b) The sequence $f(\bar{\mathbf{y}}(s))$ converges in the sense that

$$\liminf_{s\to\infty} f(\bar{\mathbf{y}}(s)) \geq \mathsf{P} - \frac{\epsilon G^2}{2} \tag{2.27a}$$

$$\textit{and} \quad \limsup_{s\to\infty} f(\bar{\mathbf{y}}(s)) \leq \mathsf{P}. \tag{2.27b}$$

Equation (2.26) asserts that the sequence $\{\bar{\mathbf{y}}(\ell)\}$ together with an associated $\{\mathring{\mathbf{p}}(\mathbf{h}; \ell)\}$ becomes asymptotically feasible. Moreover, (2.27) explicates the asymptotic suboptimality as a function of the stepsize, and the bound on the subgradient norm. Proposition 2.3 however, does not provide a way to actually find $\{\mathring{\mathbf{p}}(\mathbf{h}; \ell)\}$.

Averaging of the primal iterates is a well-appreciated method to obtain optimal primal solutions from dual subgradient methods in convex optimization [113]. Note though that the primal problem at hand is non-convex in general. Results related to Proposition 2.3 are shown in [64]. Proposition 2.3 follows in this chapter as a special case result for a more general algorithm allowing for asynchronous subgradients and suitable for online network control, elaborated next.

2.3 Subgradient Algorithm for Network Control

The algorithm in Section 2.2.2 finds the optimal operating point of (2.14) in an offline fashion. In the present section, the subgradient method is adapted so that it can be used for resource allocation during network operation.

The algorithm is motivated by Proposition 2.3 as follows. The exogenous arrival rates $a^m(\ell)$ generated by the subgradient method [cf. (2.20a)] can be used as the instantaneous rate of the traffic admitted at the transport layer at time ℓ. Then, Proposition 2.3 guarantees that the long-term average transport layer rates will be optimal. Similar observations can be made for other rates in the network.

More generally, an online algorithm with the following characteristics is desirable.

- Time is divided in slots and each subgradient iteration takes one time slot. The channel is assumed to remain invariant per slot but is allowed to vary across slots.
- Each layer maintains its set of dual variables, which are updated according to (2.22) with a constant stepsize ϵ.
- The instantaneous transmission and reception rates at the various layers are set equal to the primal iterates at that time slot, found using (2.20).
- Proposition 2.3 ensures that the long-term average rates are optimal.

For network resource allocation problems such as those described in [97], the subgradient method naturally lends itself to an online algorithm with the aforementioned properties. This approach however cannot be directly extended to the present case because the dual updates (2.22d)–(2.22e) require an expectation operation, which needs prior knowledge of the exact channel distribution function for generation of independent realizations of $\mathbf{h}$ per time slot. Furthermore, although Proposition 2.3 guarantees the existence of a sequence of feasible power variables $\mathring{\mathbf{p}}(\mathbf{h}; s)$, it is not clear if one could find them since the corresponding running averages do not necessarily converge.

Towards adapting the subgradient method for network control, recall that the subgradients $\breve{q}_{\lambda}^{iJ}$ and $\breve{q}_{\mu}^{i}$ involve the following summands that require the expectation operations [cf. (2.23d) and (2.23e)]

$$\tilde{C}_{iJ}(\ell) := \mathbb{E}\left[\sum_{f} C_{iJ}^{f}(\mathbf{p}(\mathbf{h};\ell),\mathbf{h})\right] \tag{2.28}$$

$$\tilde{P}_{i}(\ell) := \mathbb{E}\left[\sum_{f,J:(i,J)\in\mathcal{A}} p_{ij}^{f}(\mathbf{h};\ell)\right]. \tag{2.29}$$

These expectations can however be approximated by averaging over actual channel realizations. To do this, the power allocation subproblem (2.20f) must be solved repeatedly for a prescribed number of time slots, say S, while using the same Lagrange multipliers. This would then allow approximation of the $\mathbb{E}$ operations in (2.28) and (2.29) with averaging operations, performed over channel realizations at these time slots.

It is evident however, that the averaging operation not only consumes S time slots but also that the resulting subgradient is always outdated. Specifically, if the current time slot is of the form $\ell = KS+1$ with $K = 0, 1, 2, \ldots$, the most recent approximations of $\tilde{C}_{iJ}$ and $\tilde{P}_i$ available are

$$\hat{C}_{iJ}(\ell - S) = \frac{1}{S}\sum_{\kappa=\ell-S}^{\ell-1}\sum_{f} C_{iJ}^{f}(\mathbf{p}(\mathbf{h}_\kappa;\ell - S),\mathbf{h}_\kappa) \tag{2.30a}$$

$$\hat{P}_{i}(\ell - S) = \frac{1}{S}\sum_{\kappa=\ell-S}^{\ell-1}\sum_{f,J:(i,J)\in\mathcal{A}} p_{ij}^{f}(\mathbf{h}_\kappa;\ell - S). \tag{2.30b}$$

Here, the power allocations are calculated using (2.20f) with the *old* multipliers $\lambda_{iJ}(\ell - S)$ and $\mu_i(\ell - S)$. The presence of outdated subgradient summands motivates the use of an asynchronous subgradient method such as the one in [83].

Specifically, the dual updates still occur at every time slot but are allowed to use subgradients with outdated summands. Thus, $\hat{C}_{iJ}(\ell - S)$ and $\hat{P}_i(\ell - S)$ are used instead of the corresponding $\mathbb{E}[.]$ terms in (2.23d) and (2.23e) at the current time ℓ. Further, since the averaging operation consumes another S time slots, the same summands are also used for times $\ell+1$, $\ell+2, \ldots, \ell+S-1$. At time $\ell+S$, power allocations from the time slots ℓ, $\ell+1$, $\ell+S-1$ become available, and are used for calculating $\hat{C}_{iJ}(\ell)$ and $\hat{P}_i(\ell)$, which then serve as the more recent subgradient summands. Note that a subgradient summand such as $\hat{C}_{iJ}$ is at least S and at most $2S-1$ slots old.

Algorithm 2.1: Asynchronous subgradient algorithm

```
1  Initialize $\boldsymbol{\zeta}(1) = 0$ and $\hat{C}_{iJ}(1) = \hat{P}_i(1) = 0$.    Let $N$ be the maximum
   number of subgradient iterations.
2  for $\ell = 1, 2, \ldots, N$, do
3      Calculate primal iterates $a^m(\ell)$, $x_{ij}^{mt}(\ell)$, $z_{iJ}^m(\ell)$, $c_{iJ}(\ell)$, and $p_i(\ell)$
       [cf. (2.20a)–(2.20e)].
4      Calculate the optimal power allocation $\mathbf{p}(\mathbf{h}_\ell; \tau(\ell))$ by solving (2.20f) using $\mathbf{h}_\ell$ and
       $\boldsymbol{\zeta}(\tau(\ell))$.
5      Update dual iterates $\nu_i^{mt}(\ell+1)$, $\eta_{ik}^{mt}(\ell+1)$ and $\xi_{ij}(\ell+1)$ from the current primal
       iterates evaluated in Line 3 [cf. (2.22a)–(2.22c)].
6      if $\ell - \tau(\ell) = S$, then
7          Calculate $\hat{C}_{iJ}(\tau(\ell))$ and $\hat{P}_i(\tau(\ell))$ as in (2.30).
8      end
9      Update the dual iterates $\lambda_{iJ}(\ell+1)$ and $\mu_i(\ell+1)$:
           $\lambda_{iJ}(\ell+1) = \left[\lambda_{iJ}(\ell) + \epsilon(c_{iJ}(\ell) - \hat{C}_{iJ}(\tau(\ell)))\right]^+$
           $\mu_i(\ell+1) = \left[\mu_i(\ell) + \epsilon(\hat{P}_i(\tau(\ell)) - p_i(\ell))\right]^+.$
10     Network Control: Use the current iterates $a^m(\ell)$ for flow control; $x_{ij}^{mt}(\ell)$ and $z_{iJ}^m(\ell)$
       for routing and network coding; $c_{iJ}(\ell)$ for link rate control; and $\mathbf{p}(\mathbf{h}_\ell; \tau(\ell))$ for
       instantaneous power allocation.
11 end
```

The asynchronous subgradient method is summarized as Algorithm 1. The algorithm uses the function $\tau(\ell)$ which outputs the time of most recent averaging operation, that is,

$$\tau(\ell) = \max\bigl\{S\lfloor(\ell - S - 1)/S\rfloor + 1, 1\bigr\} \quad \forall \; \ell \geq 1. \tag{2.32}$$

Note that $S \leq \ell - \tau(\ell) \leq 2S - 1$. Recall also that the subgradient components $\hat{C}_{iJ}$ and $\hat{P}_i$ are evaluated only at times $\tau(\ell)$.

The following proposition gives the dual convergence result on this algorithm. Define $\bar{G}$ as the bound $\left\|[\hat{\mathbf{C}}^T \; \hat{\mathbf{P}}^T]^T\right\| \leq \bar{G}$ where $\hat{\mathbf{C}}$ and $\hat{\mathbf{P}}$ are formed by stacking the terms $\mathbb{E}\left[\sum_f C_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h})\right]$ and $\mathbb{E}\left[\sum_{f,J} p_{ij}^f(\mathbf{h})\right]$, respectively.

Proposition 2.4. *If the maximum delay of the asynchronous counterparts of physical layer updates* (2.22d) *and* (2.22e) *is* D*, then:*

a) The sequence of dual iterates $\{\boldsymbol{\zeta}(\ell)\}$ *is bounded; and*

b) The best dual value converges to D *up to a constant:*

$$\lim_{s\to\infty} \min_{1\leq\ell\leq s} \varrho(\boldsymbol{\zeta}(\ell)) \leq \mathsf{D} + \frac{\epsilon G^2}{2} + 2\epsilon D\bar{G}G. \tag{2.33}$$

Thus, the suboptimality in the asynchronous subgradient over the synchronous version is bounded by a constant proportional to $D = 2S - 1$. Consequently, the asynchronous subgradient might need a smaller stepsize (and hence, more iterations) to reach a given distance from the optimal.

The convergence of asynchronous subgradient methods for convex problems such as (2.17) has been studied in [83, Section 6] for a diminishing stepsize. Proposition 2.4 provides a complementary result for constant stepsizes.

Again, as with the synchronous version, the primal running averages also converge to within a constant from the optimal value of (2.14). This is stated formally in the next proposition.

Proposition 2.5. *If the maximum delay of the asynchronous counterparts of physical layer updates* (2.22d) *and* (2.22e) *is* D*, then:*

a) There exists a sequence $\mathring{\mathbf{p}}(\mathbf{h};s)$ *such that* $(\bar{\mathbf{y}}(s), \mathring{\mathbf{p}}(\mathbf{h};s)) \in \mathcal{B}$ *and*

$$\lim_{s\to\infty} \left\| [\mathbf{q}(\bar{\mathbf{y}}(s), \mathring{\mathbf{p}}(\mathbf{h};s))]^+ \right\| = 0. \tag{2.34}$$

b) The sequence $f(\bar{\mathbf{y}}(s))$ *converges in the following sense:*

$$\liminf_{s\to\infty} f(\bar{\mathbf{y}}(s)) \geq \mathsf{P} - \frac{\epsilon G^2}{2} - 2\epsilon D\bar{G}G \tag{2.35a}$$

$$\textit{and} \quad \limsup_{s\to\infty} f(\bar{\mathbf{y}}(s)) \leq \mathsf{P}. \tag{2.35b}$$

Note that as with the synchronous subgradient, the primal running averages are still asymptotically feasible, but the bound on their suboptimality increases by a term proportional to the delay D in the physical layer updates. Of course, all the results in Propositions 2.4 and 2.5 reduce to the corresponding results in Propositions 2.2 and 2.3 on setting $D = 0$. Interestingly, there is no similar result for primal convergence in asynchronous subgradient methods even for convex problems.

Finally, the following remarks on the online nature of the algorithm and the implementation of the Lagrangian maximizations in (2.20) are in order.

Remark 2.1. Algorithm 1 has several characteristics of an online adaptive algorithm. In particular, prior knowledge of the channel distribution is not needed in order to run the algorithm since the expectation operations are replaced by averaging over channel realizations on the fly. Likewise, running averages need not be evaluated; Proposition 2.5 ensures that the corresponding long-term averages will be near-optimal. Further, if at some time the network topology changes and the algorithm keeps running, it would be equivalent to restarting the entire algorithm with the current state as initialization. The algorithm is adaptive in this sense.

Remark 2.2. Each of the maximization operations (2.20a)–(2.20e) is easy, because it involves a single variable, concave objective, box constraints, and locally available Lagrange multipliers. The power control subproblem (2.20f) however may be hard and require centralized computation in order to obtain a (near-) optimal solution. For the conflict graph model, see [71,159] and references therein for a list of approximate algorithms. For the SINR model, solutions of (2.20f) could be based on approximation techniques in power control for digital subscriber lines (DSL)—see e.g., [64] and references therein—and efficient message passing protocols as in [170].

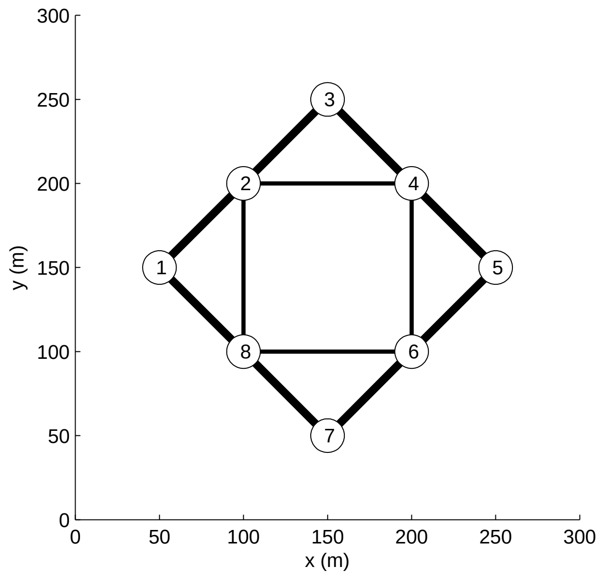


Figure 2.1: The wireless network used in the simulations. The edges indicate the neighborhood of each node. The thickness of the edges is proportional to the mean of the corresponding channel.

2.4 Numerical Tests

The asynchronous algorithm developed in Section 2.3 is simulated on the wireless network shown in Figure 2.1. The network has 8 nodes placed on a 300m $\times$ 300m area. Hyperarcs originating from node i are denoted by $(i, J) \in \mathcal{A}$ where $J \in 2^{N(i)} \setminus \emptyset$ i.e., the power set of the neighbors of i excluding the empty set. For instance, hyperarcs originating from node 1 are $(1, \{2\})$, $(1, \{8\})$ and $(1, \{2, 8\})$. The network supports the two multicast sessions $\mathcal{S}_1 = \{1, \{4, 6\}\}$ and $\mathcal{S}_2 = \{4, \{1, 7\}\}$. Table 2.4 lists the parameter values used in the simulation.

The conflict graph model of Example 1 with secondary interference constraints is used. In order to solve the power control subproblem (2.20f), we need to enumerate all possible sets of conflict free hyperarcs (cf. Example 1); these sets are called matchings. At each time slot, the aim is to find the matching that maximizes the objective function $\sum_{f,(i,J)} \gamma_{iJ}^f$. Note that since γ_{iJ}^f is a positive quantity, only maximal matchings, i.e., matchings with maximum possible cardinality, need to be considered. At each time slot, the following two steps are carried out.

S1) Find the optimal power allocation for each maximal matching. Note that the capacity of

F	2
h_{ij}^f	Exponential with mean $\bar{h}_{ij}^f = 0.1(d_{ij}/d_0)^{-2}$ for all $(i,j) \in \mathcal{G}$ and f, where $d_0 = 20$m and d_{ij} is the distance between the nodes i and j; links are reciprocal, i.e., $h_{ij}^f = h_{ji}^f$.
N_j	Noise power, evaluated using $d_{ij} = 100$m in the expression for $\bar{h}_{ij}^f$ above
$p_{\max}^f$	5 W/Hz for all f
$p_i^{\max}$	5 W/Hz for all $i \in \mathcal{N}$
$a_{\max}^m$	5 bps/Hz for all m
$a_{\min}^m$	10^{-4} bps/Hz for all m
$c_{iJ}^{\max}$	interference-free capacity obtained for each $j \in J$ via waterfilling under $\mathbb{E}\left[\sum_f p^f(h_{ij}^f)\right] \leq p_i^{\max}$ for all $i \in \mathcal{N}$
$z_{iJ}^{\max}$	$c_{iJ}^{\max}/2$ for all $(i,J) \in \mathcal{A}$
$x_{ij}^{\max}$	$z_{iJ}^{\max}/2$ for $j \in J$ and $i \in \mathcal{N}$
$U_m(a^m)$	$\ln(a^m)$ for all m
$V_i(p_i)$	$10p_i^2$ for all $i \in \mathcal{N}$

Table 2.1: Simulation parameters

an active hyperarc is a function of the power allocation over that hyperarc alone [cf. (2.3) and (2.4)]. Thus, the maximization in (2.20f) can be solved separately for each hyperarc and tone. The resulting objective [cf. (2.20g)] is a concave function in a single variable, admitting an easy waterfilling-type solution.

S2) Evaluate the objective function (2.20f) for each maximal matching and for powers found in Step 2, and choose the matching with the highest resulting objective value.

It is well known that the enumeration of hyperarc matchings requires exponential complexity [159]. Since the problem at hand is small, full enumeration is used.

Figure 2.2 shows the evolution of the utility function $f(\bar{\mathbf{y}}(s))$ and the best dual value up to the current iteration. The utility function is evaluated using the running average of the primal iterates [cf. (2.25)]. It can be seen that after a certain number of iterations, the primal and dual

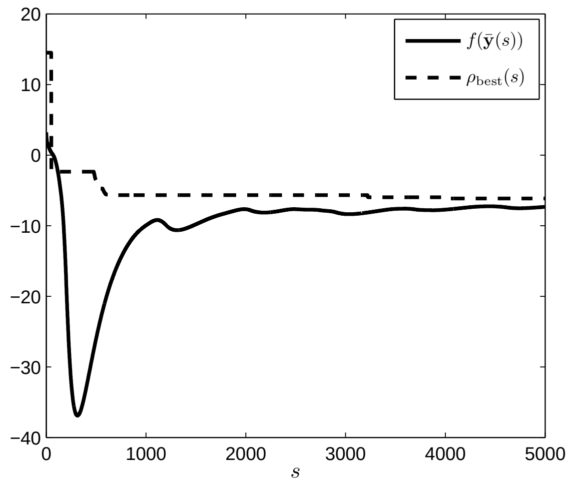


Figure 2.2: Evolution of the utility function $f(\bar{\mathbf{y}}(s))$ and best dual value $\rho_{\text{best}}(s) = \min_{\ell \leq s} \varrho(\boldsymbol{\zeta}(\ell))$ for $\epsilon = 0.15$ and $S = 50$.

values remain very close corroborating the vanishing duality gap.

Figure 2.3 shows the evolution of the utility function for different values of S. Again the utility function converges to a near-optimal value after sufficient number of iterations. Note however that the gap from the optimal dual value increases for large values of S, such as $S = 60$ (cf. Proposition 2.5).

Finally, Figure 2.4 shows the optimal values of certain optimization variables. Specifically the two subplots show all the virtual flows to given sinks for each of the multicast sessions, namely, $\{s^1 = 1, t = 6\}$ and $\{s^2 = 4, t = 7\}$, respectively. The thickness and the gray level of the edges is proportional to the magnitude of the virtual flows. It can be observed that most virtual flows are concentrated along the shorter paths between the source and the sink. Also, the radius of the circles representing the nodes is proportional to the optimal average power consumption. It can be seen that the inner nodes 2, 4, 6, and 8 consume more power than the outer ones, 1, 3, 5, and 7. This is because the inner nodes have more neighbors, and thus more opportunities to transmit. Moreover, the outer nodes are all close to their neighbors.

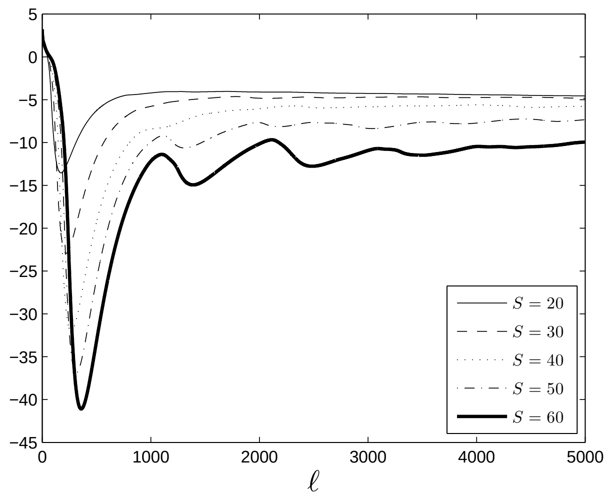


Figure 2.3: Evolution of the utility function $f(\bar{\mathbf{y}}(s))$ for different values of S with stepsize $\epsilon = 0.15$.

2.5 Conclusions

This chapter formulates a cross-layer optimization problem for multicast networks where nodes perform intra-session network coding, and operate over fading broadcast links. Zero duality gap is established, rendering layered architectures optimal.

Leveraging this result, an adaptation of the subgradient method suitable for network control is also developed. The method is asynchronous, because the physical layer returns its contribution to the subgradient vector with delay. Using the subgradient vector, primal iterates in turn dictate routing, network coding, and resource allocation. It is established that network variables, such as the long-term average rates admitted into the network layer, converge to near-optimal values, and the suboptimality bound is provided explicitly as a function of the delay in the subgradient evaluation.

2.A Strong Duality for the Networking Problem (2.14)

This appendix formulates a general version of problem (2.14), and gives results about its duality gap. Let $\mathbf{h}$ be the random channel vector in $\Omega := \mathbb{R}_+^{d_{\mathbf{h}}}$, where $\mathbb{R}_+$ denotes the nonnegative reals, and $d_{\mathbf{h}}$ the dimensionality of $\mathbf{h}$. Let $\mathcal{D}$ be the σ-field of Borel sets in Ω, and $P_{\mathbf{h}}$ the

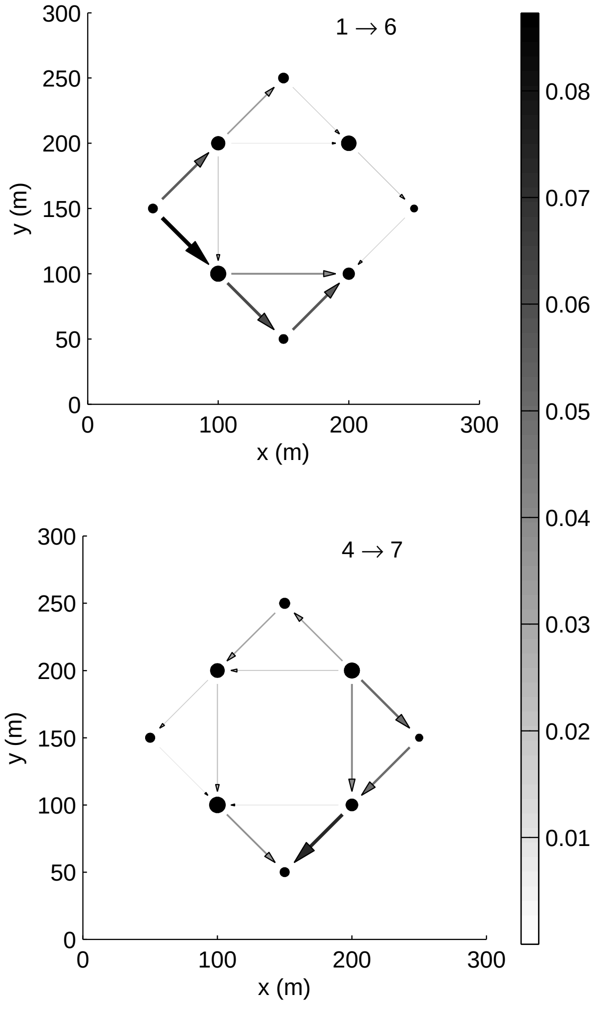


Figure 2.4: Some of the optimal primal values after 5000 iterations with $\epsilon = 0.15$ and $S = 40$. The gray level of the edges corresponds to values of virtual flows according to the color bar on the right, with units bps/Hz.

distribution of $\mathbf{h}$, which is a probability measure on $\mathcal{D}$.

As in (2.14), consider two optimization variables: the vector $\mathbf{y}$ constrained to a subset $\mathcal{B}_{\mathbf{y}}$ of the Euclidean space $\mathbb{R}^{d_{\mathbf{y}}}$; and the function $\mathbf{p} : \Omega \to \mathbb{R}^{d_{\mathbf{P}}}$ belonging to an appropriate set of functions $\mathcal{P}$. In the networking problem, the aforementioned function is the power allocation $\mathbf{p}(\mathbf{h})$, and set $\mathcal{P}$ consists of the power allocation functions satisfying instantaneous constraints, such as spectral mask or hyperarc scheduling constraints (cf. also Examples 1 and 2). Henceforth, the function variable will be denoted by $\mathbf{p}$ instead of $\mathbf{p}(\mathbf{h})$, for brevity. Let Π be a subset of $\mathbb{R}^{d_{\mathbf{P}}}$. Then $\mathcal{P}$ is defined as the set of functions taking values in Π.

$$\mathcal{P} := \{\mathbf{p} \text{ measurable} \,|\, \mathbf{p}(\mathbf{h}) \in \Pi \text{ for almost all } \mathbf{h} \in \Omega\}. \tag{2.36}$$

The network optimization problem (2.14) can be written in the general form

$$\begin{aligned} \mathsf{P} = \quad \max \quad & f(\mathbf{y}) && \text{(2.37a)} \\ \text{s. t.} \quad & \mathbf{g}(\mathbf{y}) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})] \leq \mathbf{0} && \text{(2.37b)} \\ & \mathbf{y} \in \mathcal{B}_{\mathbf{y}},\ \mathbf{p} \in \mathcal{P} && \text{(2.37c)} \end{aligned}$$

where $\mathbf{g}$ and $\mathbf{v}$ are $\mathbb{R}^d$-valued functions describing d constraints. The formulation also subsumes similar problems in the unicast routing framework such as those in [64, 139].

Evidently, problem (2.14) is a special case of (2.37). If inequalities (2.14b)–(2.14f) are rearranged to have zeros on the right hand side, function $\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})$ will simply have zeros in the entries that correspond to constraints (2.14b)–(2.14d). The function $\mathbf{q}(\mathbf{y}, \mathbf{p}(\mathbf{h}))$ defined before (2.15) equals $\mathbf{g}(\mathbf{y}) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})]$.

The following assumptions regarding (2.37) are made:

AS1. Constraint set $\mathcal{B}_{\mathbf{y}}$ is convex, closed, bounded, and in the interior of the domains of functions $f(\mathbf{y})$ and $\mathbf{g}(\mathbf{y})$. Set Π is closed, bounded, and in the interior of the domain of function $\mathbf{v}(., \mathbf{h})$ for all $\mathbf{h}$.

AS2. Function $f(\cdot)$ is concave, $\mathbf{g}(\cdot)$ is convex, and $\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})$ is integrable whenever $\mathbf{p}$ is measurable. Furthermore, there is a $\bar{G} > 0$ such that $\|\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})]\| \leq \bar{G}$, whenever $\mathbf{p} \in \mathcal{P}$.

AS3. Random vector $\mathbf{h}$ is continuous;[1] and

AS4. There exist $\mathbf{y}' \in \mathcal{B}_\mathbf{y}$ and $\mathbf{p}' \in \mathcal{P}$ such that (2.37b) holds as strict inequality (Slater constraint qualification).

Note that these assumptions are natural for the network optimization problem (2.14). Specifically, $\mathcal{B}_\mathbf{y}$ are the box constraints for variables a^m, x_{ij}^{mt}, z_{iJ}^m, c_{iJ}, and p_i; and Π gives the instantaneous power allocation constraints. The function $f(\mathbf{y})$ is selected concave and $g(\mathbf{y})$ is linear. Moreover, the entries of $\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})$ corresponding to (2.14f) are bounded because the set Π is bounded. For the same reason, the ergodic capacities $\mathbb{E}[C_{iJ}^f(\mathbf{p}(\mathbf{h}), \mathbf{h})]$ are bounded.

While (2.37) is not convex in general, it is separable [11, Section 5.1.6]. The Lagrangian (keeping constraints (2.37c) implicit) and the dual function are, respectively [cf. also (2.15) and (2.16)]

$$\mathcal{L}(\mathbf{y}, \mathbf{p}, \boldsymbol{\zeta}) = f(\mathbf{y}) - \boldsymbol{\zeta}^T \Big(\mathbf{g}(\mathbf{y}) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})]\Big) \tag{2.38}$$

$$\varrho(\boldsymbol{\zeta}) := \max_{\mathbf{y} \in \mathcal{B}_\mathbf{y},\, \mathbf{p} \in \mathcal{P}} \mathcal{L}(\mathbf{y}, \mathbf{p}, \boldsymbol{\zeta}) = \psi(\boldsymbol{\zeta}) + \phi(\boldsymbol{\zeta}). \tag{2.39}$$

where $\boldsymbol{\zeta}$ denotes the vector of Lagrange multipliers and

$$\psi(\boldsymbol{\zeta}) := \max_{\mathbf{y} \in \mathcal{B}_\mathbf{y}} \left\{ f(\mathbf{y}) - \boldsymbol{\zeta}^T \mathbf{g}(\mathbf{y}) \right\} \tag{2.40a}$$

$$\phi(\boldsymbol{\zeta}) := \max_{\mathbf{p} \in \mathcal{P}} \boldsymbol{\zeta}^T \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h}). \tag{2.40b}$$

The additive form of the dual function is a consequence of the separable structure of the Lagrangian. Further, AS1 and AS2 ensure that the domain of $\varrho(\boldsymbol{\zeta})$ is $\mathbb{R}^d$. Finally, the dual problem becomes [cf. also (2.17)]

$$\mathsf{D} = \min_{\boldsymbol{\zeta} \geq \mathbf{0}} \varrho(\boldsymbol{\zeta}). \tag{2.41}$$

As $\mathbf{p}$ varies in $\mathcal{P}$, define the range of $\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})]$ as

$$\mathcal{R} := \left\{ \mathbf{w} \in \mathbb{R}^d \,|\, \mathbf{w} = \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})] \text{ for some } \mathbf{p} \in \mathcal{P} \right\}. \tag{2.42}$$

The following lemma demonstrating the convexity of $\mathcal{R}$ plays a central role in establishing the zero duality gap of (2.37), and in the recovery of primal variables from the subgradient method.

[1]Formally, this is equivalent to saying that $P_\mathbf{h}$ is absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}_+^{d_\mathbf{h}}$. In more practical terms, it means that $\mathbf{h}$ has a probability density function without deltas.

Lemma 2.1. *If AS1-AS3 hold, then the set $\mathcal{R}$ is convex.*

The proof relies on Lyapunov's convexity theorem [16]. Recently, an extension of Lyapunov's theorem [16, Extension 1] has been applied to show zero duality gap of power control problems in DSL [98]. This extension however does not apply here, as indicated in the ensuing proof. In a related contribution [139], it is shown that the perturbation function of a problem similar to (2.37) is convex; the claim of Lemma 2.1 though is quite different.

Proof of Lemma 2.1. Let $\mathbf{r}_1$ and $\mathbf{r}_2$ denote arbitrary points in $\mathcal{R}$, and let $\alpha \in (0,1)$ be arbitrary. By the definition of $\mathcal{R}$, there are functions $\mathbf{p}_1$ and $\mathbf{p}_2$ in $\mathcal{P}$ such that

$$\mathbf{r}_1 = \int \mathbf{v}(\mathbf{p}_1(\mathbf{h}), \mathbf{h}) dP_{\mathbf{h}} \text{ and } \mathbf{r}_2 = \int \mathbf{v}(\mathbf{p}_2(\mathbf{h}), \mathbf{h}) dP_{\mathbf{h}}. \tag{2.43}$$

Now define

$$\mathbf{u}(E) := \begin{bmatrix} \int_E \mathbf{v}(\mathbf{p}_1(\mathbf{h}), \mathbf{h}) dP_{\mathbf{h}} \\ \int_E \mathbf{v}(\mathbf{p}_2(\mathbf{h}), \mathbf{h}) dP_{\mathbf{h}} \end{bmatrix}, \quad E \in \mathcal{D}. \tag{2.44}$$

The set function $\mathbf{u}(E)$ is a nonatomic vector measure on $\mathcal{D}$, because $P_{\mathbf{h}}$ is nonatomic (cf. AS3) and the functions $\mathbf{v}(\mathbf{p}_1(\mathbf{h}), \mathbf{h})$ and $\mathbf{v}(\mathbf{p}_2(\mathbf{h}), \mathbf{h})$ are integrable (cf. AS2); see [46] for definitions. Hence, Lyapunov's theorem applies to $\mathbf{u}(E)$; see also [16, Extension 1] and [139, Lemma 1].

Specifically, consider a null set Φ in $\mathcal{D}$, i.e., a set with $P_{\mathbf{h}}(\Phi) = 0$, and the whole space $\Omega \in \mathcal{D}$. It holds that $\mathbf{u}(\Phi) = \mathbf{0}$ and $\mathbf{u}(\Omega) = [\mathbf{r}_1^T, \mathbf{r}_2^T]^T$. For the chosen α, Lyapunov's theorem asserts that there exists a set $E_\alpha \in \mathcal{D}$ such that (E_α^c denotes the complement of E_α)

$$\mathbf{u}(E_\alpha) = \alpha \mathbf{u}(\Omega) + (1-\alpha)\mathbf{u}(\Phi) = \alpha \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix} \tag{2.45a}$$

$$\mathbf{u}(E_\alpha^c) = \mathbf{u}(\Omega) - \mathbf{u}(E_\alpha) = (1-\alpha) \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}. \tag{2.45b}$$

Now using these E_α and E_α^c, define

$$\mathbf{p}_\alpha(\mathbf{h}) = \begin{cases} \mathbf{p}_1(\mathbf{h}), & \mathbf{h} \in E_\alpha \\ \mathbf{p}_2(\mathbf{h}), & \mathbf{h} \in E_\alpha^c. \end{cases} \tag{2.46}$$

It is easy to show that $\mathbf{p}_\alpha(\mathbf{h}) \in \mathcal{P}$. In particular, the function $\mathbf{p}_\alpha(\mathbf{h})$ can written as $\mathbf{p}_\alpha(\mathbf{h}) = \mathbf{p}_1(\mathbf{h})\mathbb{1}_{E_\alpha} + \mathbf{p}_2(\mathbf{h})\mathbb{1}_{E_\alpha^c}$, where $\mathbb{1}_E$ is the indicator function of a set $E \in \mathcal{D}$. Hence it is measurable, as sum of measurable functions. Moreover, we have that $\mathbf{p}_\alpha(\mathbf{h}) \in \Pi$ for almost all $\mathbf{h}$, because $\mathbf{p}_1(\mathbf{h})$ and $\mathbf{p}_2(\mathbf{h})$ satisfy this property. The need to show $\mathbf{p}_\alpha(\mathbf{h}) \in \mathcal{P}$ makes [16, Extension 1] not directly applicable here.

Thus, $\mathbf{p}_\alpha(\mathbf{h}) \in \mathcal{P}$ and satisfies [cf. (2.45)]

$$\int \mathbf{v}(\mathbf{p}_\alpha(\mathbf{h}), \mathbf{h})dP_\mathbf{h} = \int_{E_\alpha} \mathbf{v}(\mathbf{p}_1(\mathbf{h}), \mathbf{h})dP_\mathbf{h} + \int_{E_\alpha^c} \mathbf{v}(\mathbf{p}_2(\mathbf{h}), \mathbf{h})dP_\mathbf{h} = \alpha\mathbf{r}_1 + (1-\alpha)\mathbf{r}_2. \tag{2.47}$$

Therefore, $\alpha\mathbf{r}_1 + (1-\alpha)\mathbf{r}_2 \in \mathcal{R}$. □

Finally, the zero duality gap result follows from Lemma 2.1 and is stated in the following proposition.

Proposition 2.6. *If AS1-AS4 hold, then problem* (2.37) *has zero duality gap, i.e.,*

$$\mathsf{P} = \mathsf{D}. \tag{2.48}$$

Furthermore, the values P *and* D *are finite, the dual problem* (2.41) *has an optimal solution, and the set of optimal solutions of* (2.41) *is bounded.*

Proof. Function $f(\mathbf{y})$ is continuous on $\mathcal{B}_\mathbf{y}$ since it is convex (cf. AS1 and AS2) [12, Prop. 1.4.6]. This, combined with the compactness of $\mathcal{B}_\mathbf{y}$, shows that the optimal primal value P is finite. Consider the set

$$\mathcal{W} := \Big\{(w_1, \ldots, w_d, u) \in \mathbb{R}^{d+1} \;\Big|\; f(\mathbf{y}) \le u, \mathbf{g}(\mathbf{y}) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})] \le \mathbf{w} \text{ for some } \mathbf{y} \in \mathcal{B}_\mathbf{y},\ \mathbf{p} \in \mathcal{P}\Big\}. \tag{2.49}$$

Using Lemma 2.1, it is easy to verify that set $\mathcal{W}$ is convex. The rest of the proof follows that of [11, Prop. 5.3.1 and 5.1.4], using the finiteness of P and Slater constraint qualification (cf. AS4).

The boundedness of the optimal dual set is a standard result for convex problems under Slater constraint qualification and finiteness of optimal primal value; see e.g., [12, Prop. 6.4.3] and [113, p. 1762]. The proof holds also in the present setup since P is finite, $\mathsf{P} = \mathsf{D}$, and AS4 holds. $\square$

2.B Dual and Primal Convergence Results

This appendix formulates the synchronous and asynchronous subgradient methods for the generic problem (2.37); and establishes the convergence claims in Propositions 2.2–2.5. Note that Propositions 2.2 and 2.3 follow from Propositions 2.4 and 2.5, respectively, upon setting the delay $D = 0$.

Starting from an arbitrary $\boldsymbol{\zeta}(1) \geq \mathbf{0}$, the subgradient iterations for (2.41) indexed by $\ell \in \mathbb{N}$ are [cf. also (2.19)]

$$\mathbf{y}(\ell) \in \arg\max_{\mathbf{y} \in \mathcal{B}_{\mathbf{y}}} \left\{ f(\mathbf{y}) - \boldsymbol{\zeta}^T(\ell)\mathbf{g}(\mathbf{y}) \right\} \tag{2.50a}$$

$$\mathbf{p}(.;\ell) \in \arg\max_{\mathbf{p} \in \mathcal{P}} \boldsymbol{\zeta}^T(\ell)\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})] \tag{2.50b}$$

$$\text{and} \qquad \boldsymbol{\zeta}(\ell+1) = [\boldsymbol{\zeta}(\ell) + \epsilon\,(\breve{\mathbf{g}}(\ell) + \breve{\mathbf{v}}(\ell))]^+ \tag{2.50c}$$

where $\breve{\mathbf{g}}(\ell)$ and $\breve{\mathbf{v}}(\ell)$ are the subgradients of functions $\psi(\boldsymbol{\zeta})$ and $\phi(\boldsymbol{\zeta})$, defined as [cf. also (2.23)]

$$\breve{\mathbf{g}}(\ell) := \mathbf{g}(\mathbf{y}(\ell)) \tag{2.51a}$$

$$\breve{\mathbf{v}}(\ell) := \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\ell), \mathbf{h})]. \tag{2.51b}$$

The iteration in (2.50c) is synchronous, because at every ℓ, both maximizations (2.50a) and (2.50b) are performed using the current Lagrange multiplier $\boldsymbol{\zeta}(\ell)$. An *asynchronous* method is also of interest and operates as follows. Here, the component $\breve{\mathbf{v}}$ of the overall subgradient used at ℓ does not necessarily correspond to the Lagrange multiplier $\boldsymbol{\zeta}(\ell)$, but to the Lagrange multiplier at a time $\tau(\ell) \leq \ell$. Noting that the maximizer in (2.50b) is $\mathbf{p}(.;\tau(\ell)))$ and the corresponding subgradient component used at ℓ is $\breve{\mathbf{v}}(\tau(\ell))$, the iteration takes the form

$$\boldsymbol{\zeta}(\ell+1) = [\boldsymbol{\zeta}(\ell) + \epsilon\,(\breve{\mathbf{g}}(\ell) + \breve{\mathbf{v}}(\tau(\ell)))]^+, \quad \ell \in \mathbb{N}. \tag{2.52}$$

The difference $\ell - \tau(\ell)$ is the delay with which the subgradient component $\breve{\mathbf{v}}$ becomes available. In Algorithm 1 for example, the delayed components are $\hat{C}_{iJ}(\tau(\ell))$ and $\hat{P}_i(\tau(\ell))$.

Next, we proceed to analyze the convergence of (2.52). Function $\mathbf{g}(\mathbf{y})$ is continuous on $\mathcal{B}_{\mathbf{y}}$ because it is convex [12, Prop. 1.4.6]. Then AS1 and AS2 imply that there exists a bound G such that for all $\mathbf{y} \in \mathcal{B}_{\mathbf{y}}$ and $\mathbf{p} \in \mathcal{P}$,

$$\|\mathbf{g}(\mathbf{y}) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}), \mathbf{h})]\| \leq G. \tag{2.53}$$

Due to this bound on the subgradient norm, algorithm (2.52) can be viewed as a special case of an approximate subgradient method [112]. We do not follow this line of analysis here though, because it does not take advantage of the source of the error in the subgradient—namely, that an old maximizer of the Lagrangian is used. Moreover, algorithm (2.52) can be viewed as a particular case of an ε-subgradient method (see [11, Section 6.3.2] for definitions). This connection is made in [83] which only deals with diminishing stepsizes; here results are proved for constant stepsizes. The following assumption is adopted for the delay $\ell - \tau(\ell)$.

AS5. There exists a finite $D \in \mathbb{N}$ such that $\ell - \tau(\ell) \leq D$ for all $\ell \in \mathbb{N}$.

AS5 holds for Algorithm 1 since the maximum delay there is $D = 2S - 1$. The following lemma collects the results needed for Propositions 2.2 and 2.4. Specifically, it characterizes the error term in the subgradient definition when $-\check{\mathbf{v}}(\tau(\ell))$ is used; and also relates successive iterates $\boldsymbol{\zeta}(\ell)$ and $\boldsymbol{\zeta}(\ell+1)$. The quantity $\bar{G}$ in the ensuing statement was defined in AS2.

Lemma 2.2. *Under AS1-AS5, the following hold for the sequence* $\{\boldsymbol{\zeta}(\ell)\}$ *generated by* (2.52) *for all* $\boldsymbol{\theta} \geq \mathbf{0}$

a) $$-\check{\mathbf{v}}^T(\tau(\ell))\,(\boldsymbol{\theta} - \boldsymbol{\zeta}(\ell)) \leq \phi(\boldsymbol{\theta}) - \phi(\boldsymbol{\zeta}(\ell)) + 2\epsilon D G \bar{G} \tag{2.54a}$$

b) $$-\left(\check{\mathbf{g}}(\ell) + \check{\mathbf{v}}(\tau(\ell))\right)^T (\boldsymbol{\theta} - \boldsymbol{\zeta}(\ell)) \leq \varrho(\boldsymbol{\theta}) - \varrho(\boldsymbol{\zeta}(\ell)) + 2\epsilon D G \bar{G} \tag{2.54b}$$

c) $$\|\boldsymbol{\zeta}(\ell+1) - \boldsymbol{\theta}\|^2 - \|\boldsymbol{\zeta}(\ell) - \boldsymbol{\theta}\|^2 \leq 2\epsilon\left[\varrho(\boldsymbol{\theta}) - \varrho(\boldsymbol{\zeta}(\ell))\right] + \epsilon^2 G^2 + 4\epsilon^2 D G \bar{G} \tag{2.54c}$$

Parts (a) and (b) of Lemma 2.2 assert that the vectors $-\check{\mathbf{v}}(\tau(\ell))$ and $-\check{\mathbf{g}}(\ell) - \check{\mathbf{v}}(\tau(\ell))$ are respectively ε-subgradients of $\phi(\boldsymbol{\zeta})$ and the dual function $\varrho(\boldsymbol{\zeta})$ at $\boldsymbol{\zeta}(\ell)$, with $\varepsilon = 2\epsilon D G \bar{G}$. Note that ε is a constant proportional to the delay D.

Proof of Lemma 2.2. a) Rewrite the left-hand side of (2.54a) as

$$-\check{\mathbf{v}}^T(\tau(\ell))\,(\boldsymbol{\theta} - \boldsymbol{\zeta}(\ell)) = -\check{\mathbf{v}}^T(\tau(\ell))\,[\boldsymbol{\theta} - \boldsymbol{\zeta}(\tau(\ell))] - \check{\mathbf{v}}^T(\tau(\ell))\,[\boldsymbol{\zeta}(\tau(\ell)) - \boldsymbol{\zeta}(\ell)]\,. \tag{2.55}$$

Applying the definition of the subgradient for $\phi(\boldsymbol{\zeta})$ at $\boldsymbol{\zeta}(\tau(\ell))$ to (2.55), it follows that

$$-\check{\mathbf{v}}^T(\tau(\ell))\,(\boldsymbol{\theta}-\boldsymbol{\zeta}(\ell)) \le \phi(\boldsymbol{\theta}) - \phi(\boldsymbol{\zeta}(\tau(\ell))) - \check{\mathbf{v}}^T(\tau(\ell))\,[\boldsymbol{\zeta}(\tau(\ell))-\boldsymbol{\zeta}(\ell)]\,. \tag{2.56}$$

Now, adding and subtracting the same terms in the right-hand side of (2.56), we obtain

$$\begin{aligned}-\check{\mathbf{v}}^T(\tau(\ell))\,(\boldsymbol{\theta}-\boldsymbol{\zeta}(\ell)) \le \phi(\boldsymbol{\theta}) - \phi(\boldsymbol{\zeta}(\ell)) + \sum_{\kappa=1}^{\ell-\tau(\ell)} \left[\phi\big(\boldsymbol{\zeta}(\tau(\ell)+\kappa)\big) - \phi\big(\boldsymbol{\zeta}(\tau(\ell)+\kappa-1)\big)\right]\\ - \sum_{\kappa=1}^{\ell-\tau(\ell)} \check{\mathbf{v}}^T(\tau(\ell))\,[\boldsymbol{\zeta}(\tau(\ell)+\kappa-1)-\boldsymbol{\zeta}(\tau(\ell)+\kappa)]\,.\end{aligned} \tag{2.57}$$

Applying the definition of the subgradient for $\phi(\boldsymbol{\zeta})$ at $\boldsymbol{\zeta}(\tau(\ell)+\kappa)$ to (2.57), it follows that

$$\begin{aligned}-\check{\mathbf{v}}^T(\tau(\ell))\,(\boldsymbol{\theta}-\boldsymbol{\zeta}(\ell)) \le \phi(\boldsymbol{\theta}) - \phi(\boldsymbol{\zeta}(\ell)) + \sum_{\kappa=1}^{\ell-\tau(\ell)} \check{\mathbf{v}}^T(\tau(\ell)+\kappa)\,[\boldsymbol{\zeta}(\tau(\ell)+\kappa-1)-\boldsymbol{\zeta}(\tau(\ell)+\kappa)]\\ - \sum_{\kappa=1}^{\ell-\tau(\ell)} \check{\mathbf{v}}^T(\tau(\ell))\,[\boldsymbol{\zeta}(\tau(\ell)+\kappa-1)-\boldsymbol{\zeta}(\tau(\ell)+\kappa)]\,.\end{aligned} \tag{2.58}$$

Using the Cauchy-Schwartz inqeuality, (2.58) becomes

$$\begin{aligned}&-\check{\mathbf{v}}^T(\tau(\ell))\,(\boldsymbol{\theta}-\boldsymbol{\zeta}(\ell)) \le \phi(\boldsymbol{\theta}) - \phi(\boldsymbol{\zeta}(\ell))\\ &\quad + \sum_{\kappa=1}^{\ell-\tau(\ell)} \left(\|\check{\mathbf{v}}(\tau(\ell)+\kappa)\| + \|\check{\mathbf{v}}(\tau(\ell))\|\right)\|\boldsymbol{\zeta}(\tau(\ell)+\kappa-1)-\boldsymbol{\zeta}(\tau(\ell)+\kappa)\|\,.\end{aligned} \tag{2.59}$$

Now, write the subgradient iteration [cf. (2.52)] at $\tau(\ell)+\kappa-1$:

$$\boldsymbol{\zeta}(\tau(\ell)+\kappa) = \left[\boldsymbol{\zeta}(\tau(\ell)+\kappa-1) + \epsilon\big(\check{\mathbf{g}}(\tau(\ell)+\kappa-1) + \check{\mathbf{v}}(\tau(\tau(\ell)+\kappa-1))\big)\right]^+. \tag{2.60}$$

Subtracting $\boldsymbol{\zeta}(\tau(\ell)+\kappa-1)$ from both sides of the latter and using the nonexpansive property of the projection [12, Prop. 2.2.1] followed by (2.53), one finds from (2.60) that

$$||\boldsymbol{\zeta}(\tau(\ell)+\kappa)-\boldsymbol{\zeta}(\tau(\ell)+\kappa-1)|| \le \epsilon\,\|\check{\mathbf{g}}(\tau(\ell)+\kappa-1)+\check{\mathbf{v}}(\tau(\tau(\ell)+\kappa-1))\| \le \epsilon G. \tag{2.61}$$

Finally, recall that $\|\check{\mathbf{v}}(\ell)\| \le \bar{G}$ for all $\ell \in \mathbb{N}$ (cf. AS2), and $\ell-\tau(\ell) \le D$ for all $\ell \in \mathbb{N}$ (cf. AS5). Applying the two aforementioned assumptions and (2.61) to (2.59), we obtain (2.54a).

b) This part follows readily from part a), using (2.39) and the definition of the subgradient for $\psi(\boldsymbol{\zeta})$ at $\boldsymbol{\zeta}(\ell)$ [cf. (2.51a)].

c) We have from (2.52) for all $\boldsymbol{\theta} \geq \mathbf{0}$ that

$$\|\boldsymbol{\zeta}(\ell+1)-\boldsymbol{\theta}\|^2 = \left\|[\boldsymbol{\zeta}(\ell)+\epsilon\left(\breve{\mathbf{g}}(\ell)+\check{\mathbf{v}}(\tau(\ell))\right)]^+-\boldsymbol{\theta}\right\|^2. \tag{2.62}$$

Due to the nonexpansive property of the projection, it follows that

$$\begin{aligned}\|\boldsymbol{\zeta}(\ell+1)-\boldsymbol{\theta}\|^2 &\leq \|\boldsymbol{\zeta}(\ell)+\epsilon\left(\breve{\mathbf{g}}(\ell)+\check{\mathbf{v}}(\tau(\ell))\right)-\boldsymbol{\theta}\|^2 \\ &= \|\boldsymbol{\zeta}(\ell)-\boldsymbol{\theta}\|^2+\epsilon^2\left\|\breve{\mathbf{g}}(\ell)+\check{\mathbf{v}}(\tau(\ell))\right\|^2 \\ &\quad + 2\epsilon\left(\breve{\mathbf{g}}(\ell)+\check{\mathbf{v}}(\tau(\ell))\right)^T\left(\boldsymbol{\zeta}(\ell)-\boldsymbol{\theta}\right).\end{aligned} \tag{2.63}$$

Introducing (2.54b) and (2.53) into (2.63), (2.54c) follows. □

The main convergence results for the synchronous and asynchronous subgradient methods are given by Propositions 2.2 and 2.4, respectively. Using Lemma 2.2, Proposition 2.4 is proved next.

Proof of Proposition 2.4. a) Let $\boldsymbol{\zeta}^*$ be an arbitrary dual solution. With $\mathbf{g}_i$ and $\mathbf{v}_i$ denoting the i-th entries of $\mathbf{g}$ and $\mathbf{v}$, respectively, define

$$\delta := \min_{1\leq i\leq d}\left\{-\mathbf{g}_i(\mathbf{y}')-\mathbb{E}[\mathbf{v}_i(\mathbf{p}'(\mathbf{h}),\mathbf{h})]\right\} \tag{2.64}$$

where $\mathbf{y}'$ and $\mathbf{p}'$ are the strictly feasible variables in AS4. Note that $\delta > 0$ due to AS4.

We show that the following relation holds for all $\ell \geq 1$:

$$\|\boldsymbol{\zeta}(\ell)-\boldsymbol{\zeta}^*\| \leq \max\left\{\|\boldsymbol{\zeta}(1)-\boldsymbol{\zeta}^*\|, \frac{1}{\delta}(\mathsf{D}-f(\mathbf{y}'))+\frac{\epsilon G^2}{2\delta}+\frac{2\epsilon DG\bar{G}}{\delta}+\|\boldsymbol{\zeta}^*\|+\epsilon G\right\}. \tag{2.65}$$

Eq. (2.65) implies that the sequence of Lagrange multipliers $\{\boldsymbol{\zeta}(\ell)\}$ is bounded, because the optimal dual set is bounded (cf. Proposition 2.6). Next, (2.65) is shown by induction. It obviously holds for $\ell = 1$. Assume it holds for some $\ell \in \mathbb{N}$. It is proved next that it holds for $\ell+1$. Two cases are considered, depending on the value of $\varrho(\boldsymbol{\zeta}(\ell))$.

Case 1: $\varrho(\boldsymbol{\zeta}(\ell)) > \mathsf{D}+\epsilon G^2/2+2\epsilon DG\bar{G}$. Then eq. (2.54c) with $\boldsymbol{\theta} = \boldsymbol{\zeta}^*$ and $\varrho(\boldsymbol{\zeta}^*) = \mathsf{D}$ becomes

$$\|\boldsymbol{\zeta}(\ell+1)-\boldsymbol{\zeta}^*\|^2 \leq \|\boldsymbol{\zeta}(\ell)-\boldsymbol{\zeta}^*\|^2 - 2\epsilon\left[\varrho(\boldsymbol{\zeta}(\ell)-\mathsf{D}-\epsilon\bar{G}^2/2-2\epsilon DG\bar{G}\right]. \tag{2.66}$$

The square-bracketed quantity in (2.66) is positive due to the assumption of Case 1. Then (2.66) implies that $||\boldsymbol{\zeta}(\ell+1)-\boldsymbol{\zeta}^*||^2 < ||\boldsymbol{\zeta}(\ell)-\boldsymbol{\zeta}^*||^2$, and the desired relation holds for $\ell+1$.

Case 2: $\varrho(\boldsymbol{\zeta}(\ell)) \leq \mathsf{D}+\epsilon G^2/2+2\epsilon D G\bar{G}$. It follows from (2.52), the nonexpansive property of the projection, the triangle inequality, and the bound (2.53) that

$$\|\boldsymbol{\zeta}(\ell+1)-\boldsymbol{\zeta}^*\| \leq \left\|\boldsymbol{\zeta}(\ell)+\epsilon\big(\check{\mathbf{g}}(t)+\check{\mathbf{v}}(\tau(\ell))\big)-\boldsymbol{\zeta}^*\right\| \tag{2.67a}$$

$$\leq \|\boldsymbol{\zeta}(\ell)\|+\|\boldsymbol{\zeta}^*\|+\epsilon G \tag{2.67b}$$

Next, a bound on $||\boldsymbol{\zeta}(\ell)||$ is developed. Specifically, it holds due to the definition of the dual function [cf. (2.39)] that

$$\begin{aligned}\varrho(\boldsymbol{\zeta}(\ell)) &= \max_{\mathbf{y}\in\mathcal{B}_{\mathbf{y}},\,\mathbf{p}\in\mathcal{P}}\big\{f(\mathbf{y})-\boldsymbol{\zeta}^T(\ell)\big(\mathbf{g}(\mathbf{y})+\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}),\mathbf{h})]\big)\big\}\\ &\geq f(\mathbf{y}')-\boldsymbol{\zeta}^T(\ell)\big(\mathbf{g}(\mathbf{y}')+\mathbb{E}[\mathbf{v}(\mathbf{p}'(\mathbf{h}),\mathbf{h})]\big).\end{aligned} \tag{2.68}$$

Rewriting the inner product in (2.68) using the entries of the corresponding vectors and substituting (2.64) into (2.68) using $\boldsymbol{\zeta}\geq\mathbf{0}$, it follows that

$$\begin{aligned}\delta\sum_{i=1}^{d}\zeta_i(\ell) &\leq -\sum_{i=1}^{d}\zeta_i^T(\ell)\big(\mathbf{g}_i(\mathbf{y}')+\mathbb{E}[\mathbf{v}_i(\mathbf{p}'(\mathbf{h}),\mathbf{h})]\big)\\ &\leq \varrho(\boldsymbol{\zeta}(\ell))-f(\mathbf{y}').\end{aligned} \tag{2.69}$$

Using $\|\boldsymbol{\zeta}(\ell)\|\leq\sum_{i=1}^d\zeta_i(\ell)$ into (2.69), the following bound is obtained:

$$\|\boldsymbol{\zeta}(\ell)\|\leq\frac{1}{\delta}(\varrho(\boldsymbol{\zeta}(\ell))-f(\mathbf{y}')). \tag{2.70}$$

Introducing (2.70) into (2.67b) and using the assumption of Case 2, the desired relation (2.65) holds for $\ell+1$.

b) Set $\boldsymbol{\theta}=\boldsymbol{\zeta}^*$ and $\varrho(\boldsymbol{\theta})=\varrho(\boldsymbol{\zeta}^*)=\mathsf{D}$ in (2.54c):

$$\|\boldsymbol{\zeta}(\ell+1)-\boldsymbol{\zeta}^*\|^2\leq\|\boldsymbol{\zeta}(\ell)-\boldsymbol{\zeta}^*\|^2+\epsilon^2G^2+4\epsilon^2DG\bar{G}+2\epsilon\left[\mathsf{D}-\varrho(\boldsymbol{\zeta}(\ell))\right]. \tag{2.71}$$

Summing the latter for $\ell=1,\ldots,s$, and introducing the quantity $\min_{1\leq\ell\leq s}\varrho(\boldsymbol{\zeta}(\ell))$, it follows that

$$\begin{aligned}\|\boldsymbol{\zeta}(\ell+1)-\boldsymbol{\zeta}^*\|^2 &\leq \|\boldsymbol{\zeta}(1)-\boldsymbol{\zeta}^*\|^2+s\epsilon^2G^2+4s\epsilon^2DG\bar{G}+2s\epsilon\mathsf{D}-2\epsilon\sum_{\ell=1}^{s}\varrho(\boldsymbol{\zeta}(\ell))\\ &\leq \|\boldsymbol{\zeta}(1)-\boldsymbol{\zeta}^*\|^2+s\epsilon^2G^2+4s\epsilon^2DG\bar{G}+2s\epsilon\mathsf{D}-2s\epsilon\min_{1\leq\ell\leq s}\varrho(\boldsymbol{\zeta}(\ell)).\end{aligned} \tag{2.72}$$

Substituting the left-hand side of (2.72) with 0, rearranging the resulting inequality, we obtain

$$0 \leq \|\boldsymbol{\zeta}(1) - \boldsymbol{\zeta}^*\|^2 + s\epsilon^2 G^2 + 4s\epsilon^2 DG\bar{G} + 2s\epsilon\mathsf{D} - 2s\epsilon \min_{1\leq\ell\leq s} \varrho(\boldsymbol{\zeta}(\ell))$$

and thus,

$$\min_{1\leq\ell\leq s} \varrho(\boldsymbol{\zeta}(\ell)) \leq \mathsf{D} + \frac{\epsilon G^2}{2} + 2\epsilon DG\bar{G} + \frac{\|\boldsymbol{\zeta}(1) - \boldsymbol{\zeta}^*\|^2}{2\epsilon s}. \tag{2.73}$$

Now, note that $\lim_{s\to\infty} \min_{1\leq\ell\leq s} \varrho(\boldsymbol{\zeta}(\ell))$ exists, because $\min_{1\leq\ell\leq s} \varrho(\boldsymbol{\zeta}(\ell))$ is monotone decreasing in s and lower-bounded by D, which is finite. Moreover, $\lim_{s\to\infty} \|\boldsymbol{\zeta}(1) - \boldsymbol{\zeta}^*\|^2/(2\epsilon s) = 0$, because $\boldsymbol{\zeta}^*$ is bounded. Thus, taking the limit as $s \to \infty$ in (2.73), yields (2.33). $\square$

Note that the sequence of Lagrange multipliers in the synchronous algorithm (2.50c) is bounded. This was shown for convex primal problems in [113, Lemma 3]. Interestingly, the proof also applies in the present case since AS1-AS4 hold and imply finite optimal $\mathsf{P} = \mathsf{D}$. (cf. Proposition 2.6) Furthermore, Proposition 2.2 for the synchronous method follows from [12, Prop. 8.2.3], [139].

Next, the convergence of primal variables through running averages is considered. The following lemma collects the intermediate results for the averaged sequence $\{\bar{\mathbf{y}}(s)\}$ [cf. (2.25)], and is used to establish convergence for the generic problem (2.37) with asynchronous subgradient updates as in (2.52). Note that $\bar{\mathbf{y}}(s) \in \mathcal{B}_{\mathbf{y}}$, $s \geq 1$, because (2.25) represents a convex combination of the points $\{\mathbf{y}(1), \ldots, \mathbf{y}(s)\}$.

Lemma 2.3. *Under AS1-AS5 with $\boldsymbol{\zeta}^*$ denoting an optimal Lagrange multiplier vector, there exists a sequence $\{\mathring{\mathbf{p}}(.;s)\}$ in $\mathcal{P}$ such that for any $s \in \mathbb{N}$, it holds that*

$$\text{a) } \left\|[\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}\left[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h};s), \mathbf{h})\right]]^+\right\| \leq \frac{\|\boldsymbol{\zeta}(s+1)\|}{\epsilon s} \tag{2.74a}$$

$$\text{b) } f(\bar{\mathbf{y}}(s)) \geq \mathsf{D} - \frac{\|\boldsymbol{\zeta}(1)\|^2}{2\epsilon s} - \frac{\epsilon G^2}{2} - 2\epsilon DG\bar{G} \tag{2.74b}$$

$$\text{c) } f(\bar{\mathbf{y}}(s)) \leq \mathsf{D} + \|\boldsymbol{\zeta}^*\| \left\|[\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h};s), \mathbf{h})]]^+\right\|. \tag{2.74c}$$

Eq. (2.74a) is an upper bound on the constraint violation, while (2.74b) and (2.74c) provide lower and upper bounds on the objective function at $\bar{\mathbf{y}}(s)$. Lemma 2.3 relies on Lemma 2.1 and the fact that the averaged sequence $\{\bar{\mathbf{y}}(s)\}$ is generated from maximizers of the Lagrangian $\{\mathbf{y}(\ell)\}$ that are *not* outdated.

Proof of Lemma 2.3. a) It follows from (2.52) that

$$\boldsymbol{\zeta}(\ell+1) \geq \boldsymbol{\zeta}(\ell) + \epsilon\left(\breve{\mathbf{g}}(\ell) + \breve{\mathbf{v}}(\tau(\ell))\right). \tag{2.75}$$

Summing (2.75) over $\ell = 1, \dots, s$, using $\boldsymbol{\zeta}(1) \geq \mathbf{0}$, it follows that

$$\epsilon \sum_{\ell=1}^{s} \breve{\mathbf{g}}(\ell) + \epsilon \sum_{\ell=1}^{s} \breve{\mathbf{v}}(\tau(\ell)) \leq \boldsymbol{\zeta}(s+1)$$

and thus,

$$\frac{1}{s}\sum_{\ell=1}^{s} \breve{\mathbf{g}}(\ell) + \frac{1}{s}\sum_{\ell=1}^{s} \breve{\mathbf{v}}(\tau(\ell)) \leq \frac{\boldsymbol{\zeta}(s+1)}{\epsilon s}. \tag{2.76}$$

Now, recall the definitions of the subgradients $\breve{\mathbf{g}}(\ell)$ and $\breve{\mathbf{v}}(\tau(\ell))$ in (2.51). Due to the convexity of $\mathbf{g}(\cdot)$, it holds that

$$\mathbf{g}(\bar{\mathbf{y}}(s)) \leq \frac{1}{s}\sum_{\ell=1}^{s} \mathbf{g}(\mathbf{y}(\ell)) = \frac{1}{s}\sum_{\ell=1}^{s} \breve{\mathbf{g}}(\ell). \tag{2.77}$$

Due to Lemma 2.1, there exists $\mathring{\mathbf{p}}(\mathbf{h}; s)$ in $\mathcal{P}$ such that

$$\mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h}; s), \mathbf{h})] = \frac{1}{s}\sum_{\ell=1}^{s} \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}; \tau(\ell)), \mathbf{h})] = \frac{1}{s}\sum_{\ell=1}^{s} \breve{\mathbf{v}}(\tau(\ell)). \tag{2.78}$$

Combining (2.76), (2.77), and (2.78), it follows that

$$\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h}; s), \mathbf{h})] \leq \frac{\boldsymbol{\zeta}(s+1)}{\epsilon s}. \tag{2.79}$$

Since $\boldsymbol{\zeta}(s+1) \geq \mathbf{0}$, (2.79) yields

$$[\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h}; s), \mathbf{h})]]^{+} \leq \frac{\boldsymbol{\zeta}(s+1)}{\epsilon s}$$

and thus,

$$\left\|[\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h}; s), \mathbf{h})]]^{+}\right\| \leq \frac{\|\boldsymbol{\zeta}(s+1)\|}{\epsilon s}. \tag{2.80}$$

b) Due to the concavity of $f(\cdot)$, it holds that $f(\bar{\mathbf{y}}(s)) \geq \frac{1}{s}\sum_{\ell=1}^{s} f(\mathbf{y}(\ell))$. Adding and subtracting the same terms to the right-hand side of the latter, we have that

$$\begin{aligned} f(\bar{\mathbf{y}}(s)) \geq{} & \frac{1}{s}\sum_{\ell=1}^{s}\left[f(\mathbf{y}(\ell)) - \boldsymbol{\zeta}^{T}(\ell)\mathbf{g}(\mathbf{y}(\ell))\right] - \frac{1}{s}\sum_{\ell=1}^{s}\boldsymbol{\zeta}^{T}(\ell)\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}; \tau(\ell)), \mathbf{h})] \\ & + \frac{1}{s}\sum_{\ell=1}^{s}\boldsymbol{\zeta}^{T}(\ell)\big(\mathbf{g}(\mathbf{y}(\ell)) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h}; \tau(\ell)), \mathbf{h})]\big). \end{aligned} \tag{2.81}$$

It holds that $f(\mathbf{y}(\ell)) - \boldsymbol{\zeta}^T(\ell)\mathbf{g}(\mathbf{y}(\ell)) = \psi(\boldsymbol{\zeta}(\ell))$ due to (2.50a) and (2.39). Using the latter into (2.81),

$$\begin{aligned} f(\bar{\mathbf{y}}(s)) \geq \frac{1}{s}\sum_{\ell=1}^{s}\Big[\psi(\boldsymbol{\zeta}(\ell)) - \boldsymbol{\zeta}^T(\ell)\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\tau(\ell)),\mathbf{h})]\Big] \\ + \frac{1}{s}\sum_{\ell=1}^{s}\boldsymbol{\zeta}^T(\ell)\big(\mathbf{g}(\mathbf{y}(\ell)) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\tau(\ell)),\mathbf{h})]\big). \end{aligned} \quad (2.82)$$

Now recall that $\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\tau(\ell)),\mathbf{h})] = \check{\mathbf{v}}(\tau(\ell))$ [cf. (2.51b)]. Thus, it holds that

$$-\zeta^T(\ell)\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\tau(\ell)),\mathbf{h})] = -\boldsymbol{\zeta}^T(\tau(\ell))\check{\mathbf{v}}(\tau(\ell)) + \check{\mathbf{v}}^T(\tau(\ell))\big[\boldsymbol{\zeta}(\tau(\ell)) - \boldsymbol{\zeta}(\ell)\big]. \quad (2.83)$$

The first term in the right-hand side of (2.83) is $\phi\big(\boldsymbol{\zeta}(\tau(\ell))\big)$ ([cf. (2.50b) and (2.39)]. The second term can be lower-bounded using Lemma 2.2(a) with $\boldsymbol{\theta} = \boldsymbol{\zeta}(\tau(\ell))$. Then, (2.83) becomes

$$-\zeta^T(\ell)\mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\tau(\ell)),\mathbf{h})] \geq \phi(\boldsymbol{\zeta}(\ell)) - 2\epsilon DG\bar{G} \quad (2.84)$$

Using (2.84) into (2.82) and $\psi(\boldsymbol{\zeta}(\ell)) + \phi(\boldsymbol{\zeta}(\ell)) = \varrho(\boldsymbol{\zeta}(\ell)) \geq \mathsf{D}$, it follows that

$$f(\bar{\mathbf{y}}(s)) \geq \mathsf{D} - 2\epsilon DG\bar{G} + \frac{1}{s}\sum_{\ell=1}^{s}\boldsymbol{\zeta}^T(\ell)\big(\mathbf{g}(\mathbf{y}(\ell)) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\tau(\ell)),\mathbf{h})]\big). \quad (2.85)$$

Moreover, it follows from (2.52) and the nonexpansive property of the projection that

$$\|\boldsymbol{\zeta}(\ell+1)\|^2 \leq \big\|\boldsymbol{\zeta}(\ell) + \epsilon\big(\mathbf{g}(\mathbf{y}(\ell)) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\tau(\ell)),\mathbf{h})]\big)\big\|^2$$

and thus,

$$\begin{aligned} \|\boldsymbol{\zeta}(\ell+1)\|^2 \leq \|\boldsymbol{\zeta}(\ell)\|^2 + 2\epsilon\boldsymbol{\zeta}^T(\ell)\left(\mathbf{g}(\mathbf{y}(\ell)) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\tau(\ell)),\mathbf{h})]\right) \\ + \epsilon^2\left\|\mathbf{g}(\mathbf{y}(\ell)) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\ell),\mathbf{h})]\right\|^2. \end{aligned} \quad (2.86)$$

Summing (2.86) for $\ell = 1,\ldots,s$, dividing by $2\epsilon s$, and introducing the bound (2.53) on the subgradient norm yield

$$\frac{1}{s}\sum_{\ell=1}^{s}\boldsymbol{\zeta}^T(\ell)\big(\mathbf{g}(\mathbf{y}(\ell)) + \mathbb{E}[\mathbf{v}(\mathbf{p}(\mathbf{h};\ell),\mathbf{h})]\big) \geq -\frac{\epsilon G^2}{2} + \frac{\|\boldsymbol{\zeta}(s+1)\|^2 - \|\boldsymbol{\zeta}(1)\|^2}{2\epsilon s}. \quad (2.87)$$

Using (2.87) into (2.85) together with $\|\boldsymbol{\zeta}(s+1)\|^2 \geq 0$, one arrives readily at (2.74b).

c) Let $\boldsymbol{\zeta}^*$ be an optimal dual solution. It holds that

$$\begin{aligned} f(\bar{\mathbf{y}}(s)) = f(\bar{\mathbf{y}}(s)) - \boldsymbol{\zeta}^{*T}\left(\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h};s),\mathbf{h})]\right) \\ + \boldsymbol{\zeta}^{*T}\left(\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h};s),\mathbf{h}]\right) \end{aligned} \quad (2.88)$$

where $\mathring{\mathbf{p}}(\mathbf{h}; s)$ was defined in part (a) [cf. (2.78)].

By the definitions of D and $\boldsymbol{\zeta}^*$ [cf. (2.41)], and the dual function [cf. (2.39)], it holds that

$$\mathsf{D} = \varrho(\boldsymbol{\zeta}^*) = \max_{\mathbf{y}\in\mathcal{B}_{\mathbf{y}}, \mathbf{p}\in\mathcal{P}} \mathcal{L}(\mathbf{y}, \mathbf{p}, \boldsymbol{\zeta}^*) \geq \mathcal{L}(\bar{\mathbf{y}}, \mathring{\mathbf{p}}, \boldsymbol{\zeta}^*). \tag{2.89}$$

Substituting the latter into (2.88), it follows that

$$f(\bar{\mathbf{y}}(s)) \leq \mathsf{D} + \boldsymbol{\zeta}^{*T}\big(\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h}; s), \mathbf{h})]\big). \tag{2.90}$$

Because $\boldsymbol{\zeta}^* \geq \mathbf{0}$ and $\boldsymbol{\theta} \leq [\boldsymbol{\theta}]^+$ for all $\boldsymbol{\theta}$, (2.90) implies that

$$f(\bar{\mathbf{y}}(s)) \leq \mathsf{D} + \boldsymbol{\zeta}^{*T}\big[\mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h}; s), \mathbf{h})]\big]^+. \tag{2.91}$$

Applying the Cauchy-Schwartz inequality to the latter, (2.74c) follows readily. □

Using Lemma 2.3, the main convergence results for the synchronous and asynchronous subgradient methods are given correspondingly by Propositions 2.3 and 2.5, after substituting

$$\mathbf{q}(\bar{\mathbf{y}}(s), \mathring{\mathbf{p}}(\mathbf{h}; s)) = \mathbf{g}(\bar{\mathbf{y}}(s)) + \mathbb{E}[\mathbf{v}(\mathring{\mathbf{p}}(\mathbf{h}; s)]. \tag{2.92}$$

Proof of Proposition 2.5. a) Take limits on both sides of (2.74a) as $s \to \infty$, and use the boundedness of $\{\boldsymbol{\zeta}(s)\}$.

b) Using $\mathsf{P} = \mathsf{D}$ and taking the $\liminf$ in (2.74b), we obtain (2.35a). Moreover, using $\mathsf{P} = \mathsf{D}$, (2.74a), the boundedness of $\|\boldsymbol{\zeta}^*\|$, and taking $\limsup$ in (2.74c), (2.35b) follows. □

Chapter 3

Cross-Layer Design of Coded Multicast with Random Access

This chapter considers joint optimization of network coding and Aloha-based medium access control (MAC) for multi-hop wireless networks. The multicast throughput with a power consumption-related penalty is maximized subject to flow conservation and MAC achievable rate constraints to obtain the optimal transmission probabilities. The relevant optimization problem is inherently non-convex and hence difficult to solve even in a centralized manner. A successive convex approximation technique is employed to obtain a Karush-Kuhn-Tucker (KKT) solution. A separable problem structure is obtained and the dual decomposition technique is adopted to develop a distributed solution. The algorithm is thus applicable to large networks, and amenable to online implementation. Numerical tests verify performance and complexity advantages of the proposed approach over existing designs. A network simulation with implementation of random linear network coding shows performance very close to the one theoretically designed.

This chapter is organized as follows. The system model and the problem statement are given in Section 3.1. The successive convex approximation algorithm is described in Section 3.2. A distributed solution and its online implementation are provided in Section 3.3. Numerical tests as well as a network simulation with suitable implementation of random linear network coding are presented in Section 3.4, followed by the conclusions in Section 3.5.

3.1 System Model and Problem Statement

3.1.1 System Model

Consider a wireless network represented by a hypergraph $(\mathcal{N}, \mathcal{A})$ with the set of nodes $\mathcal{N}$ and the set of hyperarcs $\mathcal{A}$. A hyperarc $(i, J) \in \mathcal{A}$ models the broadcast channel between node i, and the set of receivers $J \subset \mathcal{N}$. The super-set $\mathcal{J}_i$ collects all such sets of receivers $\{J|(i, J) \in \mathcal{A}\}$ for node $i \in \mathcal{N}$. The one-hop neighborhood of node i is denoted by $N(i)$ and includes all nodes belonging to at least one set $J \in \mathcal{J}_i$. The hyperarc model is very general and allows nodes to transmit at different rates and powers on each hyperarc; see e.g., [96, 97, 159]. It also subsumes point-to-point, and broadcast-only scenarios, as detailed later in Section 3.1.3.

Consider further a multicast session involving a source node $s \in \mathcal{N}$, and a set of sink nodes $\mathcal{T} \subset \mathcal{N}$. The aim is to maximize the multicast rate R at which node s can transmit the same information to all the sink nodes $t \in \mathcal{T}$. The network operates in a time-slotted fashion. The unit of R and of all other rates that will be described here is packets per slot.

For networks modeled by graphs with error-free edges, random linear network coding achieves the full multicast capacity [72]. Wireless networks, however, are error-prone and have broadcast channels that are better modeled by hyperarcs. The multicast rate region with random linear network coding for such networks is also known [97, 159] and represents the achievable rate region which can be realized by practical network coding schemes such as [27, 72, 96]. Leveraging on this characterization, the present section formulates a cross-layer optimization problem to maximize the multicast rates supported by a slotted Aloha network. To this end, a set of auxiliary variables $\{r_{ij}^{(t)}\}$ is introduced, with $r_{ij}^{(t)} \geq 0$ representing the virtual transmission rate (also called virtual flow) from node i to a neighboring node $j \in N(i)$ for sink $t \in \mathcal{T}$. Virtual flows abide by the flow conservation constraints [97]

$$\sum_{j \in N(i)} r_{ij}^{(t)} - \sum_{j : i \in N(j)} r_{ji}^{(t)} = R\mathbf{1}_{\{i=s\}} - R\mathbf{1}_{\{i=t\}}, \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{3.1}$$

where $\mathbf{1}_{\{.\}}$ is the indicator function that takes the value one when the expression inside the curly brackets is true, and zero otherwise.

Optimization solvers usually require all constraints to be expressed as inequalities. Therefore, the following relaxed version of virtual flow constraints (3.1) is used here

$$\sum_{j\in N(i)} r_{ij}^{(t)} - \sum_{j:i\in N(j)} r_{ji}^{(t)} \geq R\mathbb{1}_{\{i=s\}}, \qquad\qquad t\in\mathcal{T}, i\in\mathcal{N}\setminus\{t\}. \tag{3.2}$$

To obtain (3.2), note that in (3.1), the set of equations for $i=t$ can be omitted since they are implied by the other equations. Relaxation of the flow constraints for $i\neq t$ is then equivalent to allowing each node i to transmit at higher rate than received, which amounts to adding virtual sources at all nodes. Note, however, that sending nonzero flow from these virtual sources to the sinks can never increase R, which is the flow from s to $t\in\mathcal{T}$. Thus, even if the optimal solution has some nodes injecting extra flows, they can all be set to zero without impeding R.

3.1.2 Characterization of MAC Constraints

The MAC layer employs the slotted Aloha protocol. At every time slot, each node $i\in\mathcal{N}$ transmits on hyperarc (i,J) with probability p_{iJ} and (instantaneous) physical layer (PHY) rate c_{iJ}. The transmissions of different nodes are independent. Not all nodes in J can decode the packets received from i, because of collisions or erasures. Let $I(m)$ denote the set of nodes whose transmissions interfere with the reception at node m. Reception at node m may fail (a) due to collisions—when a node $j\in I(m)$ or m itself (half-duplex contraint) is transmitting at the same time slot—or (b) due to erasures caused by impairments of the wireless medium or jamming. Erasure means that although the link may be collision-free, the receiving end cannot decode the transmitted packets with some probability due to, e.g., fading. Occurrence of erasures is independent of collisions. To summarize, a transmission from i to m is successful when (a) no node $j\in(I(m)\cup\{m\})\backslash\{i\}$ transmits, and (b) there is no erasure on link (i,m).

Let S_{iJ}^{m} denote the event that a packet transmitted on hyperarc (i,J) is correctly decoded by node $m\in J$, and define $q_i := 1-\sum_{J\in\mathcal{J}_i} p_{iJ}$, the probability that i remains silent. Assume that erasures happen independently across links and time slots, and let $1-s_{iJm}$ be the probability of erasure on the link (i,m) of a packet transmitted at PHY rate c_{iJ}. Assuming fully backlogged queues at the link layer, so that all nodes have packets to transmit at every time slot, one can write the probability of S_{iJ}^m as

$$\Pr\left(S_{iJ}^m\right) = s_{iJm} \prod_{j\in(I(m)\cup\{m\})\setminus\{i\}} q_j \qquad m \in J, (i,J) \in \mathcal{A}. \tag{3.3}$$

Next, introduce for each $K \subset N(i)$ the probability b_{iJK} that at least one node in K receives the packets injected on the hyperarc (i, J) correctly; i.e,

$$b_{iJK} := \Pr\left(\bigcup_{m\in K} S_{iJ}^m\right), \qquad K \subset N(i), (i,J) \in \mathcal{A}. \tag{3.4}$$

It is clear from this definition that $b_{iJK} = 0$ if $J \cap K = \varnothing$. From the inclusion-exclusion principle [142, p. 6], the probability of the union of events in (3.4) can be expanded as

$$\Pr\left(\bigcup_{m\in K} S_{iJ}^m\right) = \sum_{k=1}^{|J\cap K|} \sum_{\substack{M\subset J\cap K\\ |M|=k}} (-1)^{k-1} \Pr\left(\bigcap_{m\in M} S_{iJ}^m\right), \quad K \subset N(i), (i,J) \in \mathcal{A}. \tag{3.5}$$

Define $I(M)$, for a set of nodes $M \subset \mathcal{N}$, as the set of nodes whose transmissions interfere with at least one node in M; i.e., $I(M) = \bigcup_{m\in M} I(m)$. The probability that all nodes in M decode the packet is

$$\Pr\left(\bigcap_{m\in M} S_{iJ}^m\right) = \left(\prod_{m\in M} s_{iJm}\right)\left(\prod_{j\in(I(M)\cup M)\setminus\{i\}} q_j\right), \qquad M \subset J, i \in \mathcal{N}. \tag{3.6}$$

The average rate at which packets are injected in the hyperarc (i, J) is given by $z_{iJ} := c_{iJ}p_{iJ}$. The virtual flow rate for each sink $t \in \mathcal{T}$ can be related to $\{z_{iJ}\}_{(i,J)\in\mathcal{A}}$ through the following set of inequalities [159]

$$\sum_{j\in K} r_{ij}^{(t)} \le \sum_{J\in\mathcal{J}_i} z_{iJ} b_{iJK}, \qquad K \subset N(i), i \in \mathcal{N}, t \in \mathcal{T}. \tag{3.7}$$

The right-hand side represents the rate at which packets transmitted by node i reach at least one node in K, through various hyperarcs (i, J). Combining (3.4)–(3.6), the virtual flow constraints (3.7) become

$$\sum_{j\in K} r_{ij}^{(t)} \le \sum_{J\in\mathcal{J}_i} c_{iJ}p_{iJ} \sum_{k=1}^{|J\cap K|} \sum_{\substack{M\subset J\cap K\\ |M|=k}} (-1)^{k-1} \prod_{m\in M} s_{iJm}, \quad K \subset N(i),\ i \in \mathcal{N},\ t \in \mathcal{T}. \tag{3.8}$$

3.1.3 Problem Formulation

The problem of interest is to maximize the multicast throughput R while minimizing energy consumption, subject to network coding and random access constraints. Since higher values of q_i should translate to lower energy consumption at node $i \in \mathcal{N}$, a convex, decreasing function $v_i(q_i)$ is used as a cost to penalize the energy consumption.

First, the following definition is introduced, in order to streamline the notations in (3.8)

$$\mathcal{C}_{iK} := \{(J, M, k) | J \in \mathcal{J}_i, M \subset J \cap K, k = |M|\}. \tag{3.9}$$

Also define $\mathcal{I}_{iM} := (I(M) \cup M) \setminus \{i\}$ and $s_{iJM} := \prod_{m \in M} s_{iJm}$. The overall optimization problem is formulated as follows:

$$(\mathbf{P}^0) \qquad \min_{R \geq 0, \{r_{ij}^{(t)} \geq 0\}} \sum_{i \in \mathcal{N}} v_i(q_i) - R \tag{3.10a}$$

$$\text{s. t.} \qquad \sum_{j: i \in N(j)} r_{ji}^{(t)} + R\mathbb{1}_{\{i=s\}} - \sum_{j \in N(i)} r_{ij}^{(t)} \leq 0, \quad t \in \mathcal{T}, i \in \mathcal{N} \setminus \{t\} \tag{3.10b}$$

$$\sum_{j \in K} r_{ij}^{(t)} + \sum_{(J,M,k) \in \mathcal{C}_{iK}} (-1)^k c_{iJ} p_{iJ} s_{iJM} \prod_{j \in \mathcal{I}_{iM}} q_j \leq 0, \quad K \subset N(i), i \in \mathcal{N}, t \in \mathcal{T} \tag{3.10c}$$

$$\sum_{J \in \mathcal{J}_i} p_{iJ} + q_i - 1 \leq 0, \quad i \in \mathcal{N}. \tag{3.10d}$$

Note that (3.10d) is a relaxed version of the original equality constraint $\sum_{J \in \mathcal{J}_i} p_{iJ} + q_i = 1$. If the optimal solution is such that strict inequality holds in (3.10d) for a node $i \in \mathcal{N}$, then the value of q_i can be increased without changing any p_{iJ}. This will likely decrease the probability of collisions due to node i for other nodes, thus allowing at least as much throughput as before.

Problem $(\mathbf{P}^0)$ is non-convex, because constraint (3.10c) is non-convex. A logarithmic change of variables as in [25, Section 2] does not convexify the problem either, as (3.10b) and (3.10c) both become signomial constraints. For this reason, a successive convex approximation approach is pursued in the next section to obtain a KKT optimal solution efficiently.

Remark 3.1. The problem formulation (3.10) can also be used when there are no erasures—also referred to as lossless network—by setting $s_{iJm} = 1$ for all links. This is the case when, e.g., sufficiently strong error correction codes are employed at the link layer, possibly combined with appropriately reduced rates c_{iJ}. Erasures correlated over space, e.g., due to jamming, can also be incorporated in the formulation by directly plugging in the appropriate values of s_{iJM} for each set M in (3.10c).

Before concluding this section, it is worth mentioning that the proposed model subsumes wireless networks with point-to-point, and broadcast-only transmissions. The problem formulation also becomes simpler under these special cases and is briefly outlined next.

Point-to-point Transmissions

When only point-to-point transmissions are allowed, the network can be modeled by a regular graph with edges $\mathcal{E}$ instead of hyperarcs $\mathcal{A}$. Using the set $\mathcal{J}_i = \{j|(i,j) \in \mathcal{E}\}$ in (3.10c), the new constraints become

$$r_{ik}^{(t)} - c_{ik}p_{ik}s_{ik} \prod_{j \in I(k) \cup \{k\} \setminus \{i\}} q_j \leq 0, \qquad k \in N(i), i \in \mathcal{N}, t \in \mathcal{T}. \tag{3.11}$$

where $1 - s_{ik}$ is the erasure probability on link $(i,k) \in \mathcal{E}$. The sum-of-probabilities constraint (3.10d) also simplifies to $\sum_{j \in N(i)} p_{ij} + q_i \leq 1$.

Broadcast-only Transmissions

In networks with broadcast-only transmissions, node i transmits all its packets on the hyperarc $(i, N(i))$. Such a scenario arises when each node $i \in \mathcal{N}$ can only transmit at the same PHY rate c_i to all its neighbors. In this case, transmitting on a hyperarc (i,J) such that $J \subsetneq N(i)$ does not yield any rate advantage. Under this assumption, the Aloha protocol is also simplified and at each time slot, node i only transmits on $(i, N(i))$ with probability $p_{iN(i)} = 1 - q_i$. The set $\mathcal{C}_{iK}$ is replaced here by the set

$$\bar{\mathcal{C}}_{iK} = \{(M,k)|M \subset K, k = |M|\}. \tag{3.12}$$

Defining $\mathcal{I}_{iM}^1 := (I(M) \cup M) \setminus \{i\}$ and $\mathcal{I}_{iM}^2 := I(M) \cup M$, constraint (3.10c) becomes

$$\begin{aligned}\sum_{j \in K} r_{ij}^{(t)} &\leq c_i \sum_{(M,k) \in \bar{\mathcal{C}}_{iK}} (-1)^{k+1}(1-q_i)s_{iM} \prod_{j \in \mathcal{I}_{iM}} q_j \\ &= c_i \sum_{p=1}^{2} \sum_{(M,k) \in \bar{\mathcal{C}}_{iK}} (-1)^{k+p} s_{iM} \prod_{j \in \mathcal{I}_{iM}^p} q_j, \qquad K \subset N(i), i \in \mathcal{N}, t \in \mathcal{T}\end{aligned} \tag{3.13}$$

where $s_{iM} := s_{iN(i)M}$. Note that problem (3.10) remains non-convex even under both special cases.

The broadcast-only case was also considered in [140], where a centralized algorithm was developed for small-size networks. In addition to focusing here on distributed optimization that is scalable for larger networks, characterization of the MAC constraints in (3.13) (as well as in (3.10c)) is more efficient. Specifically, the MAC constraints in [140] are captured through the variables z_{iJ}, which, in turn, are described using a sum with the number of terms growing exponentially in $|\mathcal{N}|$, whereas in (3.13), the number of terms is exponential only in $|N(i)|$.

3.2 Successive Convex Approximation

Optimization over general non-convex constraints is well known to be difficult. However, depending on the problem structure, several approximation methods are available. An option is offered by successive convex approximation, which, under certain regularity conditions, guarantees first order KKT optimality [105]. In this section, the successive convex approximation approach is applied to $(\mathbf{P}^0)$. First, the general method is reviewed. Then, it is explained how to obtain a convex approximation for the cross-layer optimization problem at hand.

3.2.1 Successive Convex Approximation Procedure

Suppose that the objective function to be minimized is convex, and the constraint set is the intersection of a set $\mathcal{H} := \{\mathbf{y} | h_i(\mathbf{y}) \leq 0, i = 1, 2, \ldots, I\}$ with a convex set $\mathcal{C}$. Functions $\{h_i(\mathbf{y})\}$ are differentiable but may be non-convex in general. The set $\mathcal{C}$ captures convex constraints, if any. The idea is to solve a sequence of surrogate problems, indexed by $\ell \in \{1, 2, \ldots\}$, where $\mathcal{H}$ is substituted per iteration ℓ by a convex set $\mathcal{H}^\ell$. Since the intersection of convex sets is a convex set [18, Section 2.3.1], the resulting optimization problems are convex. Set $\mathcal{H}^{\ell+1}$ is constructed as $\mathcal{H}^{\ell+1} := \{\mathbf{y} | \tilde{h}_i(\mathbf{y}; \mathbf{y}^\ell) \leq 0, i = 1, 2, \ldots, I\}$, where $\mathbf{y}^\ell$ is the solution of the convex approximation at the ℓ-th iteration, and $\tilde{h}_i(\mathbf{y}; \mathbf{y}^\ell)$ for each i is a differentiable convex function satisfying the following three conditions:

(c1) $h_i(\mathbf{y}) \leq \tilde{h}_i(\mathbf{y}; \mathbf{y}^\ell)$ for all $\mathbf{y} \in \mathcal{H}^{\ell+1} \cap \mathcal{C}$

(c2) $h_i(\mathbf{y}^\ell) = \tilde{h}_i(\mathbf{y}^\ell; \mathbf{y}^\ell)$; and

(c3) $\nabla h_i(\mathbf{y}^\ell) = \nabla \tilde{h}_i(\mathbf{y}^\ell; \mathbf{y}^\ell)$.

The procedure is initialized at an arbitrary feasible point $\mathbf{y}^0 \in \mathcal{H} \cap \mathcal{C}$. As shown in [105], the limit of the sequence $\{\mathbf{y}^\ell\}$ is precisely a KKT point of the original (non-convex) problem. If there are more than one non-convex functions $h(\mathbf{y})$—so there is an intersection of sets of the form $\mathcal{H}$—a convex approximating function is needed for each of them, satisfying conditions (C1)–(C3).

3.2.2 Centralized Solution

In order to apply the successive convex approximation method to $(\mathbf{P}^0)$, consider first the change of variables $\tilde{p}_{iJ} := \log p_{iJ}$ and $\tilde{q}_i := \log q_i$. The objective in (3.10a) remains convex provided that the cost function $v_i(q_i) = v_i(e^{\tilde{q}_i})$ for each i is chosen to be convex in $\tilde{q}_i$. Such a requirement is not too restrictive, as it is satisfied by a large class of useful cost functions including, e.g., $v_i(q_i) = -\ln q_i$ and $v_i(q_i) = q_i^{-\alpha}$, $\alpha > 0$. Such cost functions do not allow $q_i = 0$, or, equivalently each node remains silent with nonzero probability, which has desirable effects on fairness as well as power savings. Constraints (3.10b) are not affected by the change of variables, and hence remain convex (linear). Constraints (3.10d) become

$$\sum_{J \in \mathcal{J}_i} \exp(\tilde{p}_{iJ}) + \exp(\tilde{q}_i) - 1 \leq 0, \qquad i \in \mathcal{N} \tag{3.14}$$

which are convex.

Constraints (3.10c) become

$$\sum_{j \in K} r_{ij}^{(t)} - \sum_{(J,M,k) \in \mathcal{C}_{iK}^1} c_{iJ} s_{iJM} \exp\left(\tilde{p}_{iJ} + \sum_{j \in \mathcal{I}_{iM}} \tilde{q}_j\right) + \sum_{(J,M,k) \in \mathcal{C}_{iK}^2} c_{iJ} s_{iJM} \exp\left(\tilde{p}_{iJ} + \sum_{j \in \mathcal{I}_{iM}} \tilde{q}_j\right)$$
$$\leq 0, \quad K \subset N(i), i \in \mathcal{N}, t \in \mathcal{T} \tag{3.15}$$

where the odd-k and even-k subsets of $\mathcal{C}_{iK}$ are defined as

$$\mathcal{C}_{iK}^1 := \{(J,M,k) | (J,M,k) \in \mathcal{C}_{iK}, k \text{ odd}\} \tag{3.16}$$

$$\mathcal{C}_{iK}^2 := \{(J,M,k) | (J,M,k) \in \mathcal{C}_{iK}, k \text{ even}\}. \tag{3.17}$$

It is noted that the second summand (with its sign) in (3.15) is concave in the optimization variables, while the rest are convex. However, it is possible to upper-bound the concave terms by an affine function [18, p. 69]. Specifically, given the solution $\tilde{p}_{iJ}^{(\ell)}$ and $\tilde{q}_j^{(\ell)}$ of the ℓ-th convex approximation, (3.15) can be replaced with the following convex constraint at the $(\ell+1)$-th approximation:

$$\sum_{j\in K} r_{ij}^{(t)} - \sum_{(J,M,k)\in\mathcal{C}_{iK}^1} c_{iJ} s_{iJM} \alpha_{iJM}^{(\ell)} \left(1 + \tilde{p}_{iJ} - \tilde{p}_{iJ}^{(\ell)} + \sum_{j\in\mathcal{I}_{iM}} (\tilde{q}_j - \tilde{q}_j^{(\ell)})\right)$$
$$+ \sum_{(J,M,k)\in\mathcal{C}_{iK}^2} c_{iJ} s_{iJM} \exp\left(\tilde{p}_{iJ} + \sum_{j\in\mathcal{I}_{iM}} \tilde{q}_j\right) \leq 0, \quad K\subset N(i), i\in\mathcal{N}, t\in\mathcal{T} \quad (3.18)$$

where for $(J,M,k)\in\mathcal{C}_{iK}^1$, $K\subset N(i)$, and $i\in\mathcal{N}$, it holds that $\alpha_{iJM}^{(\ell)} := \exp\left(\tilde{p}_{iJ}^{(\ell)} + \sum_{j\in\mathcal{I}_{iM}} \tilde{q}_j^{(\ell)}\right)$. It is easily verified that the approximation introduced in (3.18) satisfies conditions (c1)–(c3).

The resulting convex optimization problem for the $(\ell+1)$-th iteration is given by

$$(\mathbf{P}_\ell^1) \qquad \min_{R\geq 0, \{r_{ij}^{(t)}\geq 0\}, \{\tilde{p}_{iJ}\leq 0\}, \{\tilde{q}_i\leq 0\}} \sum_{i\in\mathcal{N}} v_i(e^{\tilde{q}_i}) - R \qquad (3.19a)$$
$$\text{s. t.} \qquad (3.10b),\ (3.14),\ \text{and}\ (3.18) \qquad (3.19b)$$

and can be solved by generic algorithms for convex programs such as interior-point methods; see e.g., [11], [18]. Note that in the first iteration, $\{\tilde{p}_{iJ}^{(0)}, \tilde{q}_i^{(0)}\}$ must be initialized to a feasible point of the original non-convex problem $(\mathbf{P}^0)$. This can be done by selecting arbitrary values for p_{iJ} such that $\sum_{J\in\mathcal{J}_i} p_{iJ} < 1$, setting $q_i = 1 - \sum_J p_{iJ}$, and R as well as $\{r_{ij}^{(t)}\}$ to zero.

3.2.3 Implementation

The successive convex approximation procedure outlined in Section 3.2.2 can be used to solve $(\mathbf{P}^0)$ to KKT-optimality. The algorithm must be executed offline in a centralized fashion to obtain the transmission probabilities $\{p_{iJ}, q_i\}$. Using the scheme of [96], at each time slot, node i simply transmits random linear combinations of packets in its buffer on hyperarc (i,J) with rate c_{iJ} and probability p_{iJ}. However, a centralized solver may require a long time to solve each surrogate problem $(\mathbf{P}_\ell^1)$ and need several successive approximations to converge.

Algorithm 3.1 describes an online variation of previously described algorithm, which uses the probabilities $\{\exp(\tilde{p}_{iJ}^{(\ell)}), \exp(\tilde{q}_i^{(\ell)})\}$ for transmission, as and when they become available.

This is allowed since in the limit, the variables $\{\tilde{p}_{iJ}^{(\ell)}, \tilde{q}_i^{(\ell)}\}$ become KKT-optimal. The random network coding scheme, adopted from [27], ensures that the asymptotic throughput achieved is also KKT-optimal. Interestingly, the scheme does not require MAC/network-layer acknowledgments or retransmissions; only the sinks need to signal the end of each generation.

As the size of the network scales, it is of prime interest to solve $(\mathbf{P}^0)$ in a distributed manner. Moreover, it is desired that the iterative optimization is performed online so that (slow) variations in the network topology and parameters can be tracked. Towards these ends, a distributed algorithm is developed next, which also lends itself to an online implementation.

3.3 Distributed Algorithm

Solving convex network optimization problems in a distributed fashion usually involves application of problem-specific decomposition techniques. The aim is to decompose the original problem into smaller sub-problems, which can be solved by distributed processors coordinated through local message passing. A popular method is the dual decomposition technique based on Lagrangian duality, which is well-motivated when the primal problem has a separable structure [11, Section 5.1.6], [93].

Unfortunately, the convex approximation $(\mathbf{P}_\ell^1)$ is not separable. In particular, the summands in (3.18) with even k involve exponentials of the sum of the transmission probabilities of neighboring nodes. Therefore, they do not take the form of a sum of terms that depend on individual node variables. To cope with this hurdle, additional approximation is introduced first to effect a separable structure. Moreover, a set of auxiliary variables $\{R^{(t)}\}_{t\in\mathcal{T}}$ is introduced to allow decomposition of the problem to the individual sinks in $\mathcal{T}$. For simplicity, the algorithm development hereafter specializes to the broadcast-only case. However, the methodology extends straightforwardly.

3.3.1 Creating Separable Structure

As noted earlier, the distributed solution is developed here for networks with broadcast-only transmissions. Changing the variables $\tilde{q}_i := \log q_i$, and defining $\bar{\mathcal{C}}_{iK}^1$ and $\bar{\mathcal{C}}_{iK}^2$ as the odd-k and

Algorithm 3.1: Online implementation of the centralized algorithm

```
1  initialize
2      Convex approximation index $\ell = 0$
3      Current generation index $g = 1$
4      $\{\tilde{p}_{iJ}^{(0)}, \tilde{q}_i^{(0)}\}$ to arbitrary values satisfying (3.14)
5  foreach time slot do
       // Protocol operation
6      foreach node $i$ do
7          if node $i$ has packets of generation $g$ then
8              Transmit a random linear combination of packets from generation $g$ on the
               hyperarc $(i, J)$ with probability $\exp(\tilde{p}_{iJ}^{(\ell)})$
9          end
10         if packet is received at node $i$ then
11             Store packet if it is linearly independent of the packets already stored at
               node $i$.
12         end
13     end
14     if each sink $t \in \mathcal{T}$ can decode all packets of generation $g$ then
15         Flush all packets of generation $g$ from all nodes in the network
16         update $g \leftarrow g + 1$
17     end
       // Update the transmission probabilities
18     if solution $\{\tilde{p}_{iJ}^{*}, \tilde{q}_i^{*}\}$ to $(\mathbf{P}_{\ell}^{1})$ available then
19         update
20             $\tilde{p}_{iJ}^{(\ell+1)} \leftarrow \tilde{p}_{iJ}^{*}$, $\tilde{q}_i^{(\ell+1)} \leftarrow \tilde{q}_i^{*}$ for $i \in \mathcal{N}$, $(i, J) \in \mathcal{A}$
21             $\ell \leftarrow \ell + 1$
22     end
23 end
```

even-k subsets of $\bar{\mathcal{C}}_{iK}$ [cf. (3.12)], (3.13) becomes

$$\sum_{j\in K} r_{ij}^{(t)} - c_i \sum_{(M,k)\in\bar{\mathcal{C}}_{iK}^1} s_{iM} \exp\left(\sum_{j\in\mathcal{I}_{iM}^1} \tilde{q}_j\right) + c_i \sum_{(M,k)\in\bar{\mathcal{C}}_{iK}^2} s_{iM} \exp\left(\sum_{j\in\mathcal{I}_{iM}^1} \tilde{q}_j\right)$$
$$+ c_i \sum_{(M,k)\in\bar{\mathcal{C}}_{iK}^1} s_{iM} \exp\left(\sum_{j\in\mathcal{I}_{iM}^2} \tilde{q}_j\right) - c_i \sum_{(M,k)\in\bar{\mathcal{C}}_{iK}^2} s_{iM} \exp\left(\sum_{j\in\mathcal{I}_{iM}^2} \tilde{q}_j\right) \leq 0 \quad (3.20)$$

which can be expressed compactly as

$$\sum_{j\in K} r_{ij}^{(t)} - c_i \sum_{x=1}^{2}\sum_{p=1}^{2} \sum_{(M,k)\in\bar{\mathcal{C}}_{iK}^x} (-1)^{x+p} s_{iM} \exp\left(\sum_{j\in\mathcal{I}_{iM}^p} \tilde{q}_j\right) \leq 0. \quad (3.21)$$

Of the five terms in (3.20), the second and fifth terms (those corresponding to even $x+p$ in (3.21)) are non-convex and can be upper-bounded using affine functions as in the centralized solution. Thus, given $\tilde{q}_j^{(\ell)}$ at the ℓ-th iteration, the following approximations are used for $x = p \in \{1,2\}$:

$$\exp\left(\sum_{j\in\mathcal{I}_{iM}^p} \tilde{q}_j\right) \geq \alpha_{iMp}^{(\ell)}\left(1 + \sum_{j\in\mathcal{I}_{iM}^p} (\tilde{q}_j - \tilde{q}_j^{(\ell)})\right) \quad (3.22)$$

where, similar to before, $\alpha_{iMp}^{(\ell)} := \exp\left(\sum_{j\in\mathcal{I}_{iM}^p} \tilde{q}_j^{(\ell)}\right)$.

Note that the resultant affine terms are already separable. To make the remaining terms separable, another layer of approximation is applied to (3.20). The idea is to use the arithmetic-geometric inequality to upper-bound each term in the third and the fourth summations in (3.20). Specifically, it is noted that [25, p. 32]

$$\prod_{j\in\mathcal{I}_{iM}^p} \exp(\tilde{q}_j) \leq \sum_{j\in\mathcal{I}_{iM}^p} \beta_{iMjp}^{(\ell)} \exp\left(\frac{\tilde{q}_j}{\beta_{iMjp}^{(\ell)}}\right) \quad (3.23)$$

is satisfied for terms corresponding to $x = 3 - p \in \{1,2\}$, provided that $\beta_{iMjp}^{(\ell)} > 0$ and $\sum_{j\in\mathcal{I}_{iM}} \beta_{iMjp}^{(\ell)} = 1$ hold. Moreover, it can be verified that conditions (c1)–(c3) are satisfied at $\tilde{q}_j = \tilde{q}_j^{(\ell)}$, $j \in \mathcal{I}_{iM}^p$, if the approximation parameters $\{\beta_{iMjp}^{(\ell)}\}$ are chosen for $(M,k) \in \bar{\mathcal{C}}_{iK}$,

$K \subset N(i)$, $j \in \mathcal{I}_{iM}^{p}$, and $i \in \mathcal{N}$ as $\beta_{iMjp}^{(\ell)} = \frac{\tilde{q}_j^{(\ell)}}{\sum_{j' \in \mathcal{I}_{iM}^{p}} \tilde{q}_{j'}^{(\ell)}}$. Thus, (3.20) can be surrogated by

$$\begin{aligned}
&\sum_{j \in K} r_{ij}^{(t)} - c_i \sum_{\substack{x=p\in\{1,2\}\\(M,k)\in\bar{\mathcal{C}}_{iK}^{x}}} s_{iM}\alpha_{iMp}^{(\ell)}\left(1 + \sum_{j\in\mathcal{I}_{iM}^{p}}(\tilde{q}_j - \tilde{q}_j^{(\ell)})\right)\\
&+ c_i \sum_{\substack{x=3-p\in\{1,2\}\\(M,k)\in\bar{\mathcal{C}}_{iK}^{x}}} s_{iM} \sum_{j\in\mathcal{I}_{iM}^{p}} \beta_{iMjp}^{(\ell)} \exp\left(\frac{\tilde{q}_j}{\beta_{iMjp}^{(\ell)}}\right) \le 0, \quad K \subset N(i), i \in \mathcal{N}, t \in \mathcal{T} \qquad (3.24)
\end{aligned}$$

which is now separable in the per-node optimization variables $\{r_{ij}^{(t)}\}$ and $\tilde{q}_i$ for each $i \in \mathcal{N}$.

To induce per-sink decomposability of constraint (3.10b), a set of auxiliary variables $\{R^{(t)}\}_{t\in\mathcal{T}}$ is introduced, which represents the multicast rates for the individual sinks $t \in \mathcal{T}$, and additional constraints are imposed to ensure that the sinks can support the optimal R. Specifically, (3.10b) is substituted with

$$\sum_{j:i\in N(j)} r_{ji}^{(t)} + R^{(t)}\mathbf{1}_{\{i=s\}} - \sum_{j\in N(i)} r_{ij}^{(t)} \le 0, \qquad t \in \mathcal{T}, i \in \mathcal{N}\setminus\{t\} \qquad (3.25)$$

$$R - R^{(t)} \le 0, \qquad t \in \mathcal{T}. \qquad (3.26)$$

The resulting problem

$$(\mathbf{P}_\ell^2) \qquad \min_{R\ge 0, \{R^{(t)}\ge 0\}, \{r_{ij}^{(t)}\ge 0\}, \{\tilde{q}_i \le 0\}} \sum_{i\in\mathcal{N}} v_i(e^{\tilde{q}_i}) - R \qquad (3.27a)$$

$$\text{s. t.} \qquad (3.24), (3.25), \text{ and } (3.26) \qquad (3.27b)$$

is amenable to a distributed solution, as detailed next.

3.3.2 Distributed Solution via Dual Subgradient Method

The convex optimization problem (3.27) is solved here in a distributed fashion via the dual decomposition technique. Since the objective function in (3.27a) is not *strictly* convex with respect to all primal variables, the dual function may not be differentiable. Thus, the subgradient method is employed to solve the dual problem [12, Ch. 8]. The subgradient method is widely used in cross-layer optimization; see e.g., [23, 26, 93, 97] and references therein. Also, to ensure feasibility of the primal solution recovered from the dual optimal variables, primal averaging is employed [113].

Upon introducing the Lagrange multipliers $\{\lambda_{iKt} \geq 0\}$ and $\{\mu_t \geq 0\}$ to relax constraints (3.24) and (3.26), respectively, the partial Lagrangian for (3.27) is written as

$$\begin{aligned}\mathcal{L}(R, \{R^{(t)}\}, \{r_{ij}^{(t)}\}, \{\tilde{q}_i\}) = &\sum_{i\in\mathcal{N}} v_i\left(e^{\tilde{q}_i}\right) - R + \sum_{t\in\mathcal{T}} \mu_t\left(R - R^{(t)}\right)\\ &+ \sum_{\substack{K\subset N(i),\\ i\in\mathcal{N}, t\in\mathcal{T}}} \lambda_{iKt}\Bigg\{\sum_{j\in K} r_{ij}^{(t)} - c_i \sum_{\substack{x=p\in\{1,2\}\\ (M,k)\in\bar{\mathcal{C}}_{iK}^{x}}} s_{iM}\alpha_{iMp}^{(\ell)}\left(1 + \sum_{j\in\mathcal{I}_{iM}^{p}} (\tilde{q}_j - \tilde{q}_j^{(\ell)})\right)\\ &\qquad + c_i \sum_{\substack{x=3-p\in\{1,2\}\\ (M,k)\in\bar{\mathcal{C}}_{iK}^{x}}} s_{iM} \sum_{j\in\mathcal{I}_{iM}^{p}} \beta_{iMjp}^{(\ell)} \exp\left(\frac{\tilde{q}_j}{\beta_{iMjp}^{(\ell)}}\right)\Bigg\}. \end{aligned} \tag{3.28}$$

Thus, the dual function is given by

$$D(\{\lambda_{iKt}\}, \{\mu_t\}) = \min_{\substack{R\geq 0, \{R^{(t)}\geq 0\},\\ \{r_{ij}^{(t)}\geq 0\}, \{\tilde{q}_i\leq 0\}}} \mathcal{L}(R, \{R^{(t)}\}, \{r_{ij}^{(t)}\}, \{\tilde{q}_i\}), \quad \text{s. t. (3.25)} \tag{3.29}$$

and the dual problem by

$$\max_{\{\lambda_{iKt}\geq 0\}, \{\mu_t\geq 0\}} D(\{\lambda_{iKt}\}, \{\mu_t\}). \tag{3.30}$$

In particular, the dual problem is solved using the subgradient method. This is approach is popular for network optimization problems; see e.g., [93], [26] and references therein for uncoded networks, and [97] for coded networks. The separable structure is leveraged in order to decompose the problem in smaller, easier to solve tasks which map to various network control functions, such as flow control.

First, a general description of the subgradient algorithm for the dual of a convex optmization problem is given [12, Section 8.2]. Consider the standard problem of minimizing the convex function $f_0(\mathbf{y})$. Suppose the (convex) constraints are partitioned into sets of explicit constraints $\mathbf{f}_1(\mathbf{y}) \leq 0$ and implicit constraints $\mathbf{f}_2(\mathbf{y}) \leq 0$, while there may be an additional convex set constraint $\mathbf{y} \in \mathcal{Y}$. Associate Lagrange multipliers $\boldsymbol{\zeta}$ with the explicit constraints. Then, the associated Lagrangian function is

$$\mathcal{L}(\mathbf{y}, \boldsymbol{\zeta}) := f(\mathbf{y}) + \boldsymbol{\zeta}^T \mathbf{f}_1(\mathbf{y}) \tag{3.31}$$

The subgradient iterations indexed by $\tau = 0, 1, 2, \ldots$ proceed as

$$\mathbf{y}(\tau) \in \underset{\mathbf{y}:f_2(\mathbf{y})\leq 0, \mathbf{y}\in\mathcal{Y}}{\arg\min} \mathcal{L}(\mathbf{y}, \boldsymbol{\zeta}(\tau)) \tag{3.32a}$$

$$\boldsymbol{\zeta}(\tau+1) = [\boldsymbol{\zeta}(\tau) + \epsilon f_1(\mathbf{y}(\tau))]^+ \tag{3.32b}$$

where $[.]^+$ denotes the nonnegative orthant. The iterations are initialized with arbitrary $\boldsymbol{\zeta}(0) \geq \mathbf{0}$.

Remark 3.2. The choice of which constraints to explicitly relax via dual variables and which to keep implicit may affect complexity of the minimization step in (3.3.2), as well as the convergence speed of the algorithm. Specifically, if only few constraints are kept implicit, the primal solution step (3.3.2) may be simple, but the subgradient method to solve (3.30) may take long time to converge. On the other hand, keeping many constraints implicit may hinder distributed implementation of (3.3.2), as it becomes hard to exploit the separable structure. Here, inspired by [97] and [169], the virtual flow constraints (3.25) are kept implicit, which leads to a favorable trade-off between decomposability and convergence speed.

The separable structure of (3.27) allows terms in the Lagrangian function to be re-grouped according to the corresponding layers in the networking protocol. Thus, minimization of Lagrangian decomposes to per-layer sub-problems in the link layer (involving the log probabilities $\{\tilde{q}_i\}$); the network layer (involving the network coding parameters $\{R^{(t)}\}$ and $\{r_{ij}^{(t)}\}$); and the transport layer (involving the multicast rate R), each of which can be solved individually given the Lagrange multipliers. In the sequel, distributed solutions to the sub-problems are developed.

Link layer sub-problem

The link layer sub-problem can be further decomposed to the node level. Upon defining the set $\mathcal{I}_i^{-p}$ of nodes that are interfered by node i's transmission as

$$\mathcal{I}_i^{-p} := \{m \in \mathcal{N} | i \in \cup_{M\subset N(m)} \mathcal{I}_{mM}^p\} \tag{3.33}$$

the link layer sub-problem for node $i \in \mathcal{N}$ is obtained by collecting in $\mathcal{L}(\cdot)$ of (3.3.2) the terms containing $\tilde{q}_i$ (henceforth, τ denotes the iteration index of the subgradient updates to be discussed later):

$$\tilde{q}_i(\tau) \in \arg\min_{\tilde{q}_i \leq 0} \; v_i\left(e^{\tilde{q}_i}\right) - c_i \tilde{q}_i \left(\sum_{\substack{x=1 \\ p=x}}^{2} \sum_{m \in \mathcal{I}_i^{-p}} \sum_{\substack{K \subset N(m) \\ t \in \mathcal{T}}} \sum_{(M,k) \in \mathcal{C}_{mK}^x} s_{mM} \alpha_{mMp}^{(\ell)} \lambda_{mKt}(\tau) \right)$$
$$+ c_i \left(\sum_{\substack{x=1 \\ p=3-x}}^{2} \sum_{m \in \mathcal{I}_i^{-p}} \sum_{\substack{K \subset N(m) \\ t \in \mathcal{T}}} \sum_{(M,k) \in \mathcal{C}_{mK}^x} s_{mM} \beta_{mMip}^{(\ell)} \lambda_{mKt}(\tau) \exp\left(\frac{\tilde{q}_i}{\beta_{mMip}^{(\ell)}} \right) \right) \quad (3.34)$$

Network layer sub-problem

The network layer sub-problem can be further decomposed to the sink level. Thus, $R^{(t)}$ and $\{r_{ij}^{(t)}\}_{j \in N(i), i \in \mathcal{N}}$ can be updated by solving the per-sink problem for each $t \in \mathcal{T}$ given by

$$(R^t(\tau), \{r_{ij}^{(t)}(\tau)\}) \in \arg\min_{R^{(t)} \geq 0, \{r_{ij}^{(t)} \geq 0\}} \sum_{j \in N(i), i \in \mathcal{N}} r_{ij}^{(t)} \sum_{\substack{K \subset N(i) \\ K \ni j}} \lambda_{iKt}(\tau) - \mu_t(\tau) R^{(t)}$$
$$\text{s. t.} \quad \sum_{j \in N(i)} r_{ji}^{(t)} + R^{(t)} \mathbf{1}_{\{i=s\}} - \sum_{i \in N(j)} r_{ij}^{(t)} \leq 0, \quad i \in \mathcal{N} \backslash \{t\} \quad (3.35a)$$
$$r_{ij}^{(t)} \leq c_i, \quad j \in N(i), i \in \mathcal{N}. \quad (3.35b)$$

Problem (3.35) can be reduced to the standard minimum-cost flow problem by adding a virtual link from node t to node s with infinite capacity and cost $-\mu_t$ [10]. The minimum-cost flow on this graph then yields the solution to the original problem with $R^{(t)}$ given by the flow on the virtual t-s link.

The minimum-cost flow problem is a well-studied problem; see e.g., [10] for a detailed survey. Many of the algorithms available are amenable to distributed implementation, and terminate in a number of steps polynomially bounded by the number of nodes. In our case, the iterative primal updates only involve changes in the link costs. Therefore, it would be useful to choose a method that can soft-start from an available feasible solution from the previous iteration. One such method is the ϵ-relaxation method; see e.g., [10, Ch. 7], [13, Ch. 6], [63].

Transport layer sub-problem

In order to obtain the update equation for R, note first that the optimal R is necessarily upper-bounded because the per node maximum transmission rates c_i are bounded. In particular, from (3.10b) with $i = s$ and (3.10c) with $K = N(i)$, it holds that

$$R \leq \sum_{j \in N(s)} r_{sj}^{(t)} \leq c_s |\mathcal{C}_{sN(s)}| =: R_{\max}. \tag{3.36}$$

Using (3.36) as an additional constraint, the multicast rate R is updated as

$$R(\tau) \in \underset{0 \leq R \leq R_{\max}}{\arg\min} \Big(\sum_t \mu_t(\tau) - 1 \Big) R \tag{3.37}$$

which can be solved straightforwardly.

Dual update and primal recovery

Once the primal iterates $\mathbf{y}(\tau) := [\{\tilde{q}_i(\tau)\}, \{R^{(t)}(\tau)\}, \{r_{ij}^{(t)}(\tau)\}, R(\tau)]$ have been obtained, the dual variables are updated to solve (3.30). The subgradient projection method is employed, which amounts to updating the dual iterates through

$$\begin{aligned}\lambda_{iKt}(\tau+1) = \Bigg[\lambda_{iKt}(\tau) + \sigma \Bigg\{ \sum_{j \in K} r_{ij}^{(t)}(\tau) - c_i \sum_{\substack{x = p \in \{1,2\} \\ (M,k) \in \bar{\mathcal{C}}_{iK}^x}} s_{iM} \alpha_{iMp}^{(\ell)} \Bigg(1 + \sum_{j \in \mathcal{I}_{iM}^p} (\tilde{q}_j(\tau) - \tilde{q}_j^{(\ell)}) \Bigg) \\ + c_i \sum_{\substack{x = 3-p \in \{1,2\} \\ (M,k) \in \bar{\mathcal{C}}_{iK}^x}} s_{iM} \sum_{j \in \mathcal{I}_{iM}^p} \beta_{iMjp}^{(\ell)} \exp\left(\frac{\tilde{q}_j(\tau)}{\beta_{iMjp}^{(\ell)}} \right) \Bigg\} \Bigg]^+, \\ K \subset N(i), i \in \mathcal{N}, t \in \mathcal{T}\end{aligned} \tag{3.38}$$

$$\mu_t(\tau+1) = \Big[\mu_t(\tau) + \sigma \Big(R(\tau) - R^{(t)}(\tau) \Big) \Big]^+, \qquad t \in \mathcal{T} \tag{3.39}$$

where $[\cdot]^+ :\equiv \max\{0, \cdot\}$, and $\sigma > 0$ is the step size. The dual iterates can be initialized to arbitrary non-negative values. The subgradient method with a constant step size converges into a ball of the optimal dual variables, whose radius is proportional to the step size; see e.g., [12, Prop. 8.2.2] for the exact claim and the convergence rates.

Due to the lack of strict convexity, the primal iterates $\mathbf{y}(\tau)$ recovered from the dual iterates may not converge in general. Nevertheless, their running average $\bar{\mathbf{y}}(\tau) := \frac{1}{\tau} \sum_{\rho=0}^{\tau-1} \mathbf{y}(\rho)$ is asymptotically feasible, and converges to the optimum solution of $(\mathbf{P}_\ell^2)$ [113]. The running averages are then used to update $\{\tilde{q}_i^{(\ell+1)}\}$ (to evaluate $\{\alpha_{iMp}^{(\ell+1)}, \beta_{iMjp}^{(\ell+1)}\}$) for the next approximation $(\mathbf{P}_{\ell+1}^2)$ and the subgradient iterations restarted.

It is also possible to combine the subgradient and convex approximation iterations by *not* reinitializing the dual iterates when updating the values of $\{\tilde{q}_j^{(\ell)}\}$. If the surrogate problems $(\mathbf{P}_\ell^2)$ and $(\mathbf{P}_{\ell+1}^2)$ are not too different, the final dual iterates of $(\mathbf{P}_\ell^2)$ will also be near-optimal for $(\mathbf{P}_{\ell+1}^2)$. Retaining the dual iterates is therefore equivalent to "soft-starting" the dual subgradient method with near-optimal initial values. The next section builds upon this combined algorithm, and describes its distributed and online implementation.

3.3.3 Distributed and Online Protocol

The present section describes a distributed, parallel, and online implementation of the successive convex approximation algorithm. Recall from the centralized Algorithm 3.1, that it is possible to operate the network using a sequence of transmission probabilities $\{1-\exp(\tilde{q}_i^{(\ell)})\}$, converging to KKT-optimal values. In the present case, these values are provided by the combined subgradient and convex approximation algorithm outlined in Section 3.3.2. Algorithm 3.2 describes the message passing and variable updates required by the algorithm at each node $i \in \mathcal{N}$. Each subgradient iteration in Algorithm 3.2 takes up several time slots; cf. Algorithm 3.1.

Observe that the message passing required at each iteration is moderate. Specifically, node i collects primal variables $\tilde{q}_j(\tau)$ from all nodes $j \in (\cup_{M,p}\mathcal{I}_{iM}^p) \setminus \{i\} = \mathcal{I}_{iN(i)}^1$, and dual variables $\{\lambda_{mKt}\}$ from all nodes $m \in \mathcal{I}_i^{-p}$ for $p = 1, 2$. Roughly speaking, these quantities pertain to the two-hop neighborhood of node i. Further, the source needs to solve (3.35) at each iteration and for each sink (in parallel), using an asynchronous, distributed method such as ϵ-relaxation. Finally, the convex approximation parameters $\{\alpha_{iJM}^{(\ell)}, \beta_{iM}^{(\ell)}\}$ at node i depend on $\tilde{q}_j^{(\ell)}$ for $j \in \mathcal{I}_{iN(i)}^1$. These variables are anyway made known to node i for the purpose of dual updates. Overall, if each node has at most d neighbors, it exchanges $O(2^d)$ variables with each of its $O(d^2)$ two-hop neighbors, per subgradient iteration. Each node also exchanges $O(d)$ variables per ϵ-relaxation iterations. Finally, the storage requirement for each node is $O(2^d d^2)$ variables.

In general, the subgradient method does not specify a stopping criterion, and it is customary to use a fixed number of iterations. Alternatively, the subgradient algorithm can stop when the primal averages converge and remain unchanged for several iterations. In the present case, a

Algorithm 3.2: Distributed and online algorithm for node i

```
1  maintain variables
2      $\alpha_{iMp}^{(\ell)}$ and $\beta_{iMjp}^{(\ell)}$ for $M \subset N(i)$, $j \in \mathcal{I}_{iM}^{p}$, $p = 1, 2$
3      $\tilde{q}_i(\tau)$, $r_{ij}^{(t)}(\tau)$ for $j \in N(i)$, and $\lambda_{iKt}(\tau)$ for $K \subset N(i)$, $t \in \mathcal{T}$, $i \neq t$
4      if node $i$ is source then $R(\tau)$, $R^{(t)}(\tau)$, $\mu_t(\tau)$ for $t \in \mathcal{T}$
5  initialize
6      probabilities $\tilde{q}_i^{(0)}$ and evaluate $\alpha_{iMp}^{(0)}$, $\beta_{iMjp}^{(0)}$, $\lambda_{iKt}(1) = 0$ for $K \subset N(i)$, $t \in \mathcal{T}$, $i \neq t$
7      successive convex approximation index $\ell = 0$, running averages $\bar{\tilde{q}}_i(0) = 0$, and
       $\tau_0 = 0$
8      if node $i$ is source then $\mu_t(1) = 0$, for $t \in \mathcal{T}$
9  foreach $\tau = 1, 2, \ldots$ do
10     collect$\{\lambda_{mKt}(\tau)\}$ from nodes $m \in \mathcal{I}_i^{-p}$, $p = 1, 2$ and $\{\tilde{q}_j(\tau)\}$ from nodes
       $j \in \mathcal{I}_{iN(i)}^{1}$
11     update
12         primal iterates $\tilde{q}_i(\tau)$ and $r_{ij}^{(t)}(\tau)$ [cf. (3.34) and (3.35)]
13         dual iterates $\lambda_{iKt}(\tau + 1)$ [cf. (3.38)]
14         running average $\bar{\tilde{q}}_i(\tau) \leftarrow \frac{\tau - \tau_\ell - 1}{\tau - \tau_\ell} \bar{\tilde{q}}_i(\tau - 1) + \frac{1}{\tau - \tau_\ell} \tilde{q}_i(\tau)$
15     if node $i$ is the source then
16         primal iterates $R(\tau)$ and $R^{(t)}(\tau)$ [cf. (3.37) and (3.35)]
17         dual iterates $\mu_t(\tau + 1)$ [cf. (3.39)]
18     end
19     if subgradient iterations have converged or maximum iterations reached then
20         update $\tilde{q}_i^{(\ell+1)} \leftarrow \bar{\tilde{q}}_i(\tau)$ and evaluate $\alpha_{iMp}^{(\ell+1)}$ and $\beta_{iMjp}^{(\ell+1)}$
21         update $\ell \leftarrow \ell + 1$
22         reinitialize running average $\bar{\tilde{q}}_i(\tau) \leftarrow 0$, and set $\tau_\ell = \tau$
23     end
24 end
```

more sophisticated stopping criterion can also be employed. After a fixed number of iterations, each node can use the current values of $\bar{\tilde{q}}_i$ to calculate the maximum achievable throughput of the original problem $(\mathbf{P}^0)$ (specialized to broadcast case). Recall that given the probabilities, this is a linear program. If this throughput turns out to be better than the throughput of the previous convex approximation, convergence is declared and $\{\tilde{q}_j^{(\ell)}\}$ values are updated. Otherwise, the iterations continue till a prespecified maximum number.

The use of subgradient algorithm offers some flexibility in the choice of the time-scale of iterations. It is not necessary to wait for convergence of the subgradient method for updating the transmission probabilities. Indeed, the running averages $\{1 - \exp(\bar{\tilde{q}}_i(\tau))\}$ can also be used as transmission probabilities at intermediate iterations, since these converge to $\{1 - \exp(\tilde{q}_j^{(\ell)})\}$, which in turn converge to the KKT-optimal probabilities. Before concluding, a remark about alternative distributed solutions is due.

Remark 3.3. The convex problem formulated in Section 3.3.1 can also be solved by the augmented Langrangian method [143, Section 6.4.3], as an alternative to the dual subgradient approach. Note that application of this method typically makes the problem non-separable, making the dual decomposition not readily applicable. Nevertheless, it is possible to have a distributed implementation of the method, using the techniques in [13, Section 3.4].

3.4 Numerical Results

Numerical tests are performed for the centralized and distributed algorithms proposed in Section 3.2 and 3.3. Related algorithms from [140] and [160] are compared as benchmarks.

3.4.1 Simulation Set-up

Random networks are generated using the MAX-DPA algorithm [117], which generates graphs by placing nodes one by one, while respecting certain maximum-degree and node proximity constraints so as to simulate a realistic ad hoc network. The algorithm parameters are chosen to be $d = 3$, $d_{\max} = 6$, and $d_0 = 0.2$ (see [117]), and the nodes are placed in a square area with the average node density 1. The erasure probability for a pair of neighboring nodes i

# of nodes	Heurist.	Central.	B. & B.	Orth. Schedul.
7	0.1883	0.3103	0.3138	0.3144
8	0.1762	0.2736	0.2782	0.2667
9	0.1664	0.2584	0.2647	0.2498
10	0.1632	0.2354	0.2426	0.2214
20	0.1242	0.1890	–	0.1263
40	0.1110	0.1615	–	0.0776

Table 3.1: Average optimized throughput.

and j separated by distance d_{ij} is given by $1 - \exp(-d_{ij}^2/4)$, assuming Rayleigh fading. The transmission rate c_i is assumed to be unity for all nodes $i \in \mathcal{N}$. The multicast session is chosen so that the leftmost node is the source and the two rightmost nodes are the sink nodes. To compare with the existing algorithms on an equal footing, the broadcast-only scenario is considered and only the throughput is maximized while $\{v_i(q_i)\}$ are set to zero.

3.4.2 Centralized Algorithm

Table 3.4.2 gives the maximum throughput achieved with different schemes, averaged over 100 random network realizations. Four different methods are compared: the heuristic method from [160], the proposed method (centralized version), the branch-and-bound method from [140], and the orthogonal scheduling from [97] with only one transmitting node per time slot. The proposed centralized algorithm is initialized by considering a set of 20 randomly chosen probabilities q_i, and picking the one that yields the maximum value of R (which can be easily obtained by solving a linear program). It can be seen that for small-size networks, where the branch-and-bound algorithm runs in a reasonable time, the average throughput of the proposed centralized algorithm is close to the global optimum. The suboptimality is due to possible convergence of the algorithm to a KKT point.

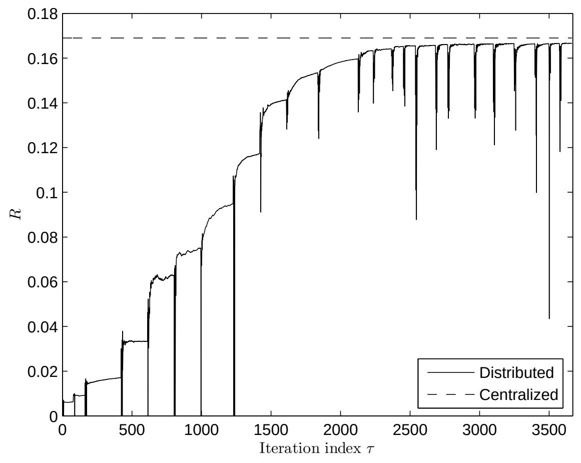


Figure 3.1: Evolution of the end-to-end throughput R in the subgradient method with step size $\sigma = 0.5$ for the first surrogate problem ($\ell = 0$) and $\sigma = 0.1$ thereafter. The vertical lines result from the fact that the primal averages are refreshed whenever the value of ℓ is advanced. Therefore the solution obtained from the next few subgradient iterations is of poor quality and gives low values of R. However, the network throughput depends only on the access probabilities at the instants when the subgradient iterations converge.

3.4.3 Distributed Algorithm

Evolution of the Subgradient Method

Algorithm 3.2 is simulated on a randomly generated network with 40 nodes. The initial point was chosen again as in Section 3.4.2. Figure 3.1 shows the evolution of the throughput achieved with the running average $\{\bar{\tilde{q}}_i\}$ (which is close but not exactly equal to the running average $\bar{R}$) across the subgradient iterations and successive convex approximations. Recall that the running averages are refreshed when the convex approximation is updated. Interestingly, the throughput converges to a near-optimal value in very few convex approximation iterations.

Online Implementation

Algorithm 3.2 is implemented with the random network coding scheme of [27] on a simple dynamic network. The network is shown in Figure 3.2 and initially consists of all nodes except

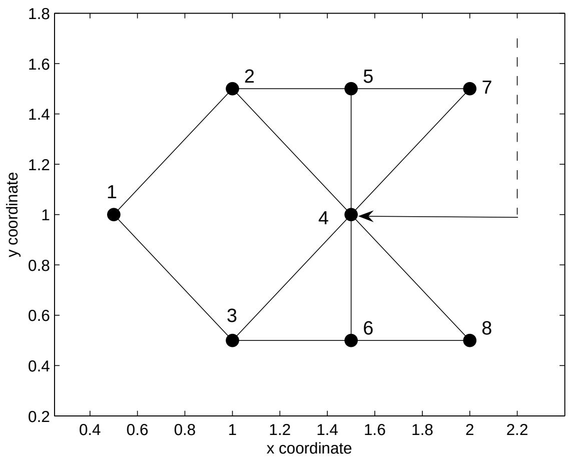


Figure 3.2: Dynamic network used for simulation. Node 4 joins the network at time slot 4×10^4.

node 4. The aim is to multicast packets from source node 1 to sink nodes 7 and 8. The network is simulated for 7×10^4 time slots, and node 4 joins the network at time slot 3×10^4 and starts transmitting with arbitrary probability.

The network coding scheme is implemented using a generation size of 100 packets, and field size 2^8. The source is infinitely backlogged, i.e., there are generations waiting to be transmitted at all time slots. It is assumed that an end-to-end network error-correction code is employed; see e.g., [8] and references therein. Consequently, the sinks are required to collect only 90 linearly independent packets for each generation. This implies that the uncoded throughput is 90% of the value obtained from the centralized solution. The subgradient algorithm runs in parallel with the network protocol, and updates the transmission probabilities every 10^3 time slots.

Figure 3.3 shows the evolution of the per-generation throughput of the system, represented by dots. The per-generation throughput is $R_g := 90/T_g$, where T_g is the time-difference (measured in time slots) between transmission of the first packet of generation g and the reception of 90 linearly independent packets of generation g at all sinks. The solid curve R_{avg} represents the moving average of R_g for 10 previously received generations. Finally, the dotted line (R_{opt}) shows the 90% of the KKT-optimal value of R obtained by running the centralized algorithm.

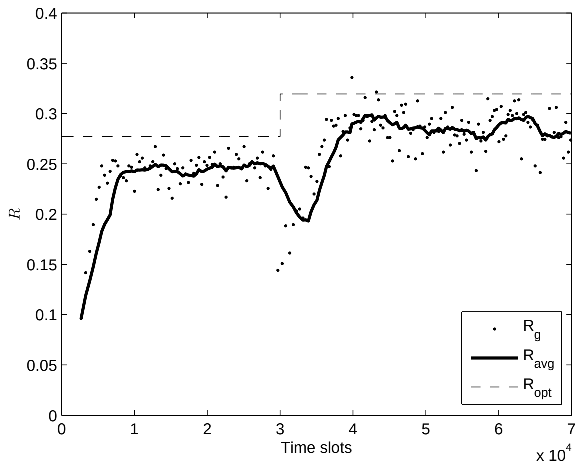


Figure 3.3: Evolution of the R values. A dot at a given time slot represents the throughput of the generation that is received at that time slot. Since generations are transmitted serially, the moving average of the per-generation throughput represents the throughput achieved over several generations.

It can be observed that the per-generation throughput is low in the beginning as all nodes start transmitting at suboptimal access probabilities. The throughput improves as the subgradient iterations evolve, but decreases again when node 4 joins the network. This is because when node 4 enters, it also starts transmitting at an arbitrary probability, and interferes with reception at other nodes. Eventually though, the subgradient iterations evolve to a new optimum, and the throughput increases again. Intuitively, node 4 helps by providing more paths for packets being multicast to nodes 7 and 8, and therefore the overall throughput is higher than before. The remaining gap between the centralized solution and achieved throughput is because of the overhead inherent to the network coding scheme. This gap can be reduced by using a larger generation size or more sophisticated schemes (such as generation-interleaving [27]) at the expense of increased end-to-end packet delay.

3.5 Conclusion

The problem of joint optimization of network coding and Aloha-based MAC for multi-hop wireless networks was considered. The multicast throughput with a power consumption-related penalty was maximized subject to flow conservation and MAC achievable rate constraints to obtain the optimal transmission probabilities. The relevant optimization problem turns out to be non-convex and hence difficult to solve even in a centralized manner. A successive convex approximation technique was employed to obtain a Karush-Kuhn-Tucker solution. The idea was also extended to create a separable structure in the problem, and the dual decomposition technique is applied to derive a distributed solution. The algorithm is thus applicable for large networks, and amenable to online implementation. The numerical tests verify performance and complexity advantages of the proposed approach over existing designs. A network simulation with implementation of random linear network coding shows performance very close to the theoretical.

Chapter 4

Cross-Layer Design of Coded Multicast under Delay Constraints

This chapter deals with network-coded multicast for real-time and streaming-media applications where packets have explicit expiration deadlines. Most of the popular network coding approaches require asymptotically large block-lengths, thereby incurring long decoding delays. The present chapter introduces a joint scheduling and network coding design that aims to maximize the average throughput while respecting the packet deadlines. The novel approach relies on a time-unwrapped graph expansion in order to construct the network codes. The resultant algorithm draws from the well-known augmenting-path algorithm, and is both distributed as well as scalable. For networks with primary interference, a lower-bound on the worst-case performance of the algorithm is provided. The associated optimization problem is also analyzed from an integer programming perspective, and a set of valid inequalities is derived to obtain an upper bound.

Related Work

The design of joint-scheduling and network coding (JS-NC) schemes with delay constraints has not been addressed in literature. Several works though, have analyzed the delay performance gains of network coding in single-hop scenarios [51,76,90]. Extension to multi-hop networks is non-trivial since the presence of scheduling constraints significantly complicates the solution.

A related problem in the context of wired networks has been analyzed in [50]. One major difference however is that [50] considers bit-by-bit transmission and network coding. This results in a problem similar to that of determining the minimum finite field size for the given network. In packet networks though, field size is usually not the bottleneck.

Several heuristic network coding schemes in media streaming applications are also available; see [34, 118, 149] and references therein. These do not design network coding jointly with scheduling constraints, and focus primarily on implementation issues. Instead, the focus here is on joint designs and performance guarantees.

Recently, there has been an attempt to reduce queuing delays in back-pressure methods by modifying the Lyapunov function [19]. This may also result in reducing queuing delay in network coding schemes that employ back-pressure. However, most of these methods also require large block-lengths, thereby rendering decoding delay a challenging bottleneck.

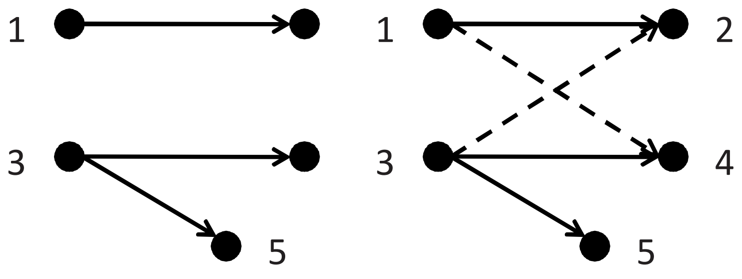
The organization of this chapter is as follows. Section 4.1 proposes a periodic version of the JS-NC problem, which is then used in Section 4.3 to derive a constant-factor approximate, augmenting-path algorithm. For networks with primary interference constraints, Section 4.4 analyzes the JS-NC design problem from an integer programming perspective. Finally, Section 4.5 presents simulated tests and Section 4.6 concludes the chapter.

4.1 System Model

Consider a wireless network represented by a directed acyclic graph $G = (V, E)$, with V denoting the set of nodes and E the set of edges. The set E consists of tuples (u, v) denoting the two nodes that the edge connects. The network supports a multicast session consisting of a source node $s \in V$ that intends to transmit a packet-stream to each of the sink nodes $T \subset V$.

Linear network coding is performed at intermediate nodes, which allows them to linearly combine and forward received packets. A block-network coding model is assumed, wherein the packet stream is parsed into blocks before transmission. Subsequently, only packets belonging to the same block are allowed to be mixed. The sinks also decode the packets in a block-wise fashion; that is, upon receiving linear combinations of the packets belonging to each block.

The network operates in a time-slotted fashion, where one time slot carries one packet.



(a) Primary Interference (b) Secondary Interference

Figure 4.1: The key difference between PI and SI constraints. Under the PI constraint, nodes 2 and 4 can simultaneously receive from transmitters 1 and 3. Under SI constraints however, the two transmitters interfere with reception at nodes 2 and 4, and should not be scheduled at the same time. Node 5 can receive from node 3 in both cases.

The *deadline constraint* dictates that the sinks must be able to decode a block within D time slots of the transmission of the *first* packet from that block by the source. Further, the wireless interface imposes the following *scheduling constraints*:

SC1. The nodes adhere to a half-duplex operational mode; and

SC2. The nodes experience interference of either (a) primary; or, (b) secondary nature.

The half-duplex constraint SC1 prevents a node from transmitting and receiving in the same time-slot. The primary interference (PI) constraint SC2(a), holds for orthogonal (i.e., channelized) access, e.g., via spreading codes or frequency division multiplexing. SC2(a) allows each node to receive from at most one neighboring node per time slot; see e.g., [39]. The secondary interference (SI) constraint SC2(b) imposes additional restrictions: two links $(u_1, v_1) \in E$ and $(u_2, v_2) \in E$ cannot be used for transmission in the same time slot if either $(u_1, v_2) \in E$ or $(u_2, v_1) \in E$; see e.g., [71]. Clearly, broadcast is allowed for both PI and SI types of constraints. Figure 4.1 shows the key difference between the PI and SI constraints.

The aim is to find the maximum multicast throughput, given here by the rate (packets per time slot) at which the source transmits packets that reach all the sinks within the stipulated deadline. In this JS-NC framework, both the time slots at which each node transmits as well as the linear combinations it uses to code must be designed.

Throughput optimization with JS-NC design is well-known to be difficult even without deadline constraint [144, 159]. Some approximate JS-NC designs are described in [71], [39] but cannot be extended to the deadlined case as they rely on using network codes with large block lengths, and consequently incur long decoding delays. This consideration motivates the following operational assumption.

AS6. The source begins transmitting the next block of packets only after the previous block has been decoded at all sinks.

Together with the deadline constraints, AS1 implies that each block of packets stays in the network for at most D time slots. As a result, the goal reduces to that of finding the JS-NC design maximizing the number of packets that can be multicast to the sinks within the first D slots. The schedule and network code can both be reused for subsequent blocks, which also makes the transmission and reception patterns of the nodes periodic with finite period D. The next section describes a time-unwrapping technique used for solving this simplified problem.

Before concluding this section, a few remarks on the assumptions are due.

Remark 4.1. The block-decoding assumption at the sink nodes is not necessarily throughput optimal. This is because the sinks begin decoding only upon receiving the linear combinations corresponding to the entire block, resulting in long waiting-times for the first few packets of the block (i.e., decoding delay). An alternative is to use infinite block-length convolutional network code designs that allow for sequential decoding at the sinks [49]. However, designing infinite block-length codes that satisfy the deadline constraints is known to be difficult even in wired networks [50].

The use of only a single block per period, as implied by AS1, incurs an overhead. This is because the source is not allowed to transmit the next block until the last packet of the current block has been received at each sink. However, as shown in Section 4.3, a solution obtained with this assumption, can be converted into a pipelined solution, that significantly reduces this overhead.

Remark 4.2. Compared to a back-pressure approach, the JS-NC design is not dynamic, i.e., the scheduling and network coding decisions are not made on a per-packet basis and do not depend on the instantaneous channel conditions. Back-pressure schemes however, are well-known to exhibit poor delay performance [19]. Further, most dynamic JS-NC algorithms require large block-lengths, resulting in prohibitive decoding delays [74]. On the other hand, the static JS-NC design proposed here offers flexibility to operate with a specified deadline D. The channel-oblivious nature of the design also makes it simpler, and easier to distribute relative to the dynamic designs.

Varying channel conditions always result in packets getting dropped or erased even in the absence of collisions. In delay-critical applications, it may not be possible to recover the lost packets at all, due to the extra time required for the sink to send feedback, and the source to re-transmit. To a certain extent, erasures can be handled through classical forward error correction codes that are applied at the source node. Alternatively, specialized random network codes are available to correct packet-erasures; see e.g., [178], [85] and references therein. In cases when the number of erasures becomes too large, partial recovery may be acceptable, and can be provided through the use of priority-encoding and transmission (PET) [27]. Several practical PET designs have been proposed in the context of network coding for video applications [137, 152].

4.2 Time-Unwrapping and Network Code Design

Under AS1, the goal is to find the JS-NC design allowing the source to multicast the maximum number of packets to each sink within D slots. This section introduces the idea of "time-unwrapping" of a graph as a tool for JS-NC design. Time-unwrapping has been employed by time-slotted networks, e.g., to solve the quickest-flow problem [20], and in the context of network code designs over wired networks [1, 50]. As the name suggests, a time-unwrapped graph can be used to represent the entire transmission and coding schedule for a given number of time slots on a single graph. The proposed construction is similar but here it must adhere also to the scheduling constraints SC1–SC2. Specifically, each node is first split into several functional subnodes, namely receiver-, combiner- and transmitter-subnodes, before being replicated. The entire procedure proceeds in these steps.

(U1) Each node v is split into receiver-, combiner- and transmitter-subnodes, and replicated

D times. The subnodes corresponding to the k-th time slot are denoted by $v^r(k)$, $v^c(k)$ and $v^t(k)$, respectively.

(U2) A directed edge (u, v) in the original graph is replaced by D directed edges $(u^t(1), v^r(1))$, $(u^t(2), v^r(2))$, and so on.

(U3) Since packets received in the current time slot are only available for transmission in the subsequent time slots, a subnode $v^r(k)$ is only connected to subnodes $v^c(k+1)$, …, $v^c(D)$.

(U4) Each combiner-subnode $v^c(k)$ is connected to its corresponding transmitter-subnode $v^t(k)$.

(U5) Finally, the source node s is modeled as a "wired" source-subnode s^ν connected to D transmitter-subnodes $s^t(1)$, $s^t(2)$, …, $s^t(D)$, i.e., s has no receiver- and combiner-subnodes.

(U6) Similarly, the set of sink nodes T is modeled by a corresponding set of "wired" sink-subnodes $\mathcal{T}$. Each wired sink-subnode $t_i^\nu \in \mathcal{T}$, for $i = 1, 2, \ldots, |T|$, receives from D receiver-subnodes $t_i^r(1)$, $t_i^r(2)$, …, $t_i^r(D)$.

Figure 4.2 shows a time-unwrapped node. The overall time-unwrapped graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ denoting the set of nodes and $\mathcal{E}$ the set of edges. Further, transmission on an edge of the form $(u^t(\ell), v^r(\ell)) \in \mathcal{E}$ corresponds to transmission on the edge $(u, v) \in E$ at time slot ℓ. Similarly, two transmissions on G, that violate SC1–SC2, give rise to a set of so-termed *conflicting* edges in $\mathcal{G}$. Thus, the entire operation (i.e., reception, combination, and transmission of packets) of the wireless network G over D time slots can be described using the time-invariant graph $\mathcal{G}$.

Given a generic time-invariant graph, any network code design algorithm takes as input the sets of edge-disjoint paths from the source to each of the sinks. The additional constraint in the present case is that the edges on these paths must not conflict with each other. Given a set of μ edge-disjoint, non-conflicting, $s^\nu - t^\nu$ paths in $\mathcal{G}$ for each sink $t^\nu \in \mathcal{T}$, an network code can be designed using one of the methods in [78]. These algorithms return the coefficients used

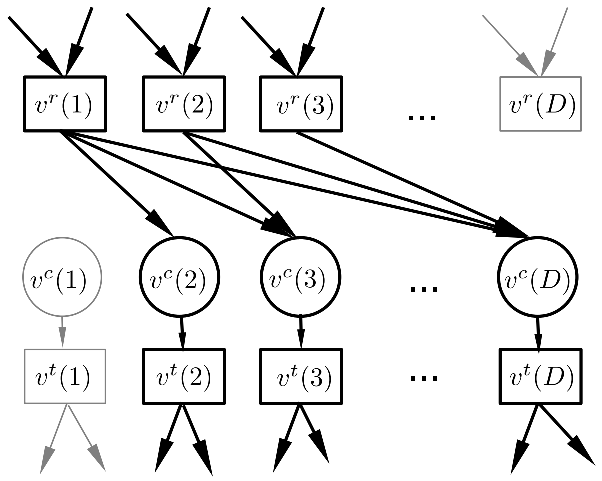


Figure 4.2: A time-unwrapped node. Note that the first combiner- and transmitter-subnodes, and the D-th receiver-subnode are redundant.

at each edge $e \in \mathcal{E}$ for linearly combining the μ packets. Formally, the vector of coefficients for edge e, referred to as the global encoding kernel, is given by $f(e) \in \mathbb{F}_q^\mu$, where $\mathbb{F}_q$ is the finite field of alphabet size q [175]. These global encoding kernels can be obtained using deterministic or randomized algorithms as those described in [78] and [73]. Indeed, if the field size $q \geq |T|$, randomly drawn kernels $\{f_e\}$ suffice with high probability, and will be used henceforth.

The linear combination provided for an edge $(u^t(\ell), v^r(\ell))$ may then be used on the edge $(u, v) \in E$ at time slot ℓ. All other edges are internal to the nodes and are only used to determine which packets need to be combined per time slot. The overall network code is therefore a list of global encoding kernels of the form $f'(e, \ell)$ for each $e \in E$ and $\ell = 1, 2, \ldots, D$. For convenience, the schedule at the ℓ-th time slot will be denoted using a graph $G_\ell = (V, E_\ell)$, where $(u, v) \in E_\ell$ if and only if the edge $(u^t(\ell), v^r(\ell)) \in \mathcal{E}$ carries a non-zero encoding vector. The overall network coding operation can therefore be viewed as the sequence of graphs namely, $G_1, G_2, \ldots G_D; G_1, G_2 \ldots, G_D; G_1, \ldots$.

It should now be clear that the multicast throughput can be maximized by finding the largest possible value of μ such that there are as many non-conflicting, edge-disjoint, augmenting paths from the source s^ν to each sink $t_i^\nu \in \mathcal{T}$, and this is the focus of the next section. But before pursuing this direction, a few remarks pertaining to the time-unwrapping procedure are in order.

Remark 4.3. It can be seen that the proposed time-unwrapped graph itself takes care of the scheduling constraints partially. For instance, the graph does not allow any path from traversing through both $v^r(\ell)$ and $v^t(\ell)$. A packet received at time slot ℓ can only be sent at a later time-slot. Similarly, the combiner-subnodes allow only one packet to be transmitted/broadcast per-time slot.

Remark 4.4. Since each packet traverses at most a single hop per-time slot, some transmitter and receiver-subnodes may be redundant. Examples include the first transmitter and combiner, and the last receiver-subnodes since only the source transmits in the first time slot, and only the sink receives in the last slot. These nodes can be removed to reduce algorithm complexity.

4.3 An Augmenting Path Approach

This section develops a greedy augmenting path (GAP) algorithm for maximizing the value of μ, the number of edge-disjoint, non-conflicting, augmenting paths from source s^ν to each sink $t^\nu \in \mathcal{T}$. A worst-case performance bound on the performance of GAP algorithm for PI networks is also established. The proposed algorithm can be viewed as an extension of the well-known Edmond-Karp algorithm [32, Ch. 26] for the wireless setting considered here.

In order to describe the GAP algorithm in detail, some graph-theoretic notions are introduced. A *flow* is an assignment of $\mathbb{F}_2$ (i.e., 0-1) values to the edges of the graph. A *valid flow* is one satisfying the flow conservation constraints; i.e., the total flow on the incoming and outgoing edges of a node should be the same. A *unit valid flow* is the assignment of 1s along an *augmenting path*, defined as any directed source-sink path. Given a flow, the *residual graph* is obtained by reversing the direction of all edges with unit values.

Finding the maximum number of edge-disjoint augmenting paths, is equivalent to finding the maximum number of unit valid flows (the max-flow problem). The Edmond-Karp(EK) max-flow algorithm proceeds as follows:

(EK0) Initialize flow values on all edges of graph $\mathcal{G}$ to zero;

(EK1) Find the shortest augmenting (source-sink) path using e.g., Dijkstra's algorithm [32, Ch. 24];

(EK2) Increment the flow values along the path found in EK1;

(EK3) Set $\mathcal{G}$ equal to the residual graph; and go back to EK1.

The idea of finding edge-disjoint augmenting paths via (EK0)–(EK3) can also be extended to the wireless setting, albeit with a modification. Specifically, after obtaining an augmenting path $\mathcal{P}$ in EK1, all other edges that conflict with any of the edges $e \in \mathcal{P}$ must be deleted from the residual graph obtained in EK3. This ensures that the augmenting paths found across iterations do not conflict with each other. Since edges are only being removed, any set of non-conflicting augmenting paths is also a feasible solution to the wired case.

In a nutshell, while repeating EK1–EK3, constraints SC1–SC2 can be respected by deleting conflicting edges till no more augmenting paths can be found. The modified EK algorithm however may not always find all the edge-disjoint augmenting paths because [32, Lemma 26.2] no longer applies. Intuitively, once an edge is deleted to obey SC1–SC2, those augmenting paths that could contain it are not present in the output of the modified EK algorithm. On the other hand, the fact that EK1 explores *shortest* augmenting paths helps to reduce the number of conflicting edges deleted.

Further modifications of the EK algorithm are needed for extension to the case with multiple sinks; see Algorithm 4.1. In this case, the algorithm maintains $|T|$ copies $\mathcal{G}_1$, $\mathcal{G}_2$, $\ldots$, $\mathcal{G}_{|T|}$ of the graph $\mathcal{G}$, one per sink. The modified EK algorithm is run per $\mathcal{G}_t$, except that conflicting edges are deleted from *all* copies $\{\mathcal{G}_t\}_{t=1}^{|T|}$. The set of edge-disjoint augmenting paths for each sink t consist of edges with unit flow values in $\mathcal{G}_t$. The *overall* flow on $\mathcal{G}$ can be obtained by assigning unit flows to those edges in $\mathcal{G}$ which have unit flows on one of the graph copies $\mathcal{G}_t$.

The list of conflicting edges to be deleted depends on the interference model used. Let, I_v (O_v) denote the set of incoming (outgoing) edges to the node $v \in \mathcal{V}$. For the PI model, the set of edges deleted at each inner iteration of Algorithm 4.1 is as follows.

(P1) For every receiver-subnode $v^r(j) \in \mathcal{P}^{(t)}$,

(a) delete edge $(v^c(j), v^t(j))$; and

Algorithm 4.1: Greedy augmenting path (GAP) algorithm

```
1 Initialization: Create copies $\{\mathcal{G}_t\}_{t=1}^{|T|}$ of the time-unwrapped graph $\mathcal{G}$. Initialize flow
  values on all edges for each graph $\mathcal{G}_t$ to zero. Set the number of edge-disjoint
  augmenting paths $\mu = 0$.
2 repeat
3     for $t$ = 1, 2, . . ., $|T|$ do
4         Find the shortest augmenting $s^\nu - t^\nu$ path $\mathcal{P}^{(t)}$ on the graph $\mathcal{G}_t$.
5         Remove edges conflicting with edges in $\mathcal{P}^{(t)}$ from all graphs $\mathcal{G}_1, \ldots, \mathcal{G}_{|T|}$.
6     end
7     Increment a unit valid flow and reverse the edges along the augmenting paths $\mathcal{P}^{(t)}$
      for each graph $\mathcal{G}_t$. Increment $\mu$ by one.
8 until an $s^\nu - t^\nu$ path can be found
```

(b) delete edge $e \in I_{v^r(j)}$ if $e \notin \mathcal{P}^{(t)}$.

(P2) For every transmitter-subnode $v^t(k) \in \mathcal{P}^{(t)}$, delete all edges $e \in I_{v^r(k)}$.

Edges are deleted in P1(a) and P2 to prevent violation of the half-duplex constraint SC1; while those deleted in P1(b) prevent violation of SC2(a).

The list of edges to be deleted in the SI constraint SC2(b) is slightly more extensive.

(S1) For every receiver-subnode $v^r(j) \in \mathcal{P}^{(t)}$,

(a) delete edge $(v^c(j), v^t(j))$;

(b) delete edge $e \in O_u$, where $(u, v^r(j)) \in \mathcal{E}$ and $u \notin \mathcal{P}^{(t)}$; and

(c) delete edge $e \in O_{u^t(j)}$ where $(v, u) \in E$.

(S2) For every transmitter-subnode $v^t(k) \in \mathcal{P}^{(t)}$,

(a) delete all edges $e \in I_{v^r(k)}$;

(b) delete all edges $e \in O_w$, where $w \notin \mathcal{P}^{(t)}$ such that $(w, u) \in \mathcal{E}$ and $(v^t(k), u) \in \mathcal{E}$ for some node u and $e \neq (v^t(k), u)$; and

(c) delete edge $e \in I_{u^{r(k)}}$, where $(u, v) \in E$.

As with the PI model, S1(a) and S2(a) take care of the half-duplex constraints. However, unlike the PI model, correct reception at a node v under SI is only ensured if all its neighboring nodes are silent. The edges corresponding to these cases are listed in S1(b), S1(c), S2(b), and S2(c). Note that Algorithm 4.1 can be extended to include more general interference models by appropriately modifying these steps.

It is worth stressing that only the original graph $\mathcal{G}$, and not the residual graph copies, are used while determining the edges to be deleted. Thus, Algorithm 4.1 outputs edge-disjoint augmenting paths for each sink as argued. Likewise, Algorithm 4.1 does not eliminate the possibility of choosing augmenting paths that delete a large number of edges. The use of shortest augmenting paths however reduces this possibility. The next subsection provides further improvements by appropriately modifying the shortest-path finder employed by Algorithm 4.1.

4.3.1 GAP Enhancements

The first part of this subsection describes a *pipelined* approach to multicast that reduces the overhead caused due to AS1 [cf. Remark 4.1]. Pipelining alters AS1 by allowing the source to multicast more than one block of packets per D time slots. The second part describes an earliest-shortest path (ESP) algorithm, for use in Algorithm 4.1 which improves throughput by reducing the number of deleted edges. In addition, the ESP algorithm enables development of worst-case bounds for the performance of Algorithm 4.1.

Pipelined Multicast

As observed in Remark 4.1, AS1 results in an overhead by allowing only one block of packets per D time slots. This yields an overall throughput of μ/D. The throughput can be improved if consecutive blocks are allowed to overlap while ensuring that they do not interfere with each other. This is effected through pipelining, which allows the source to begin transmitting the next block of packets as soon as all its neighbors have finished transmitting the current one.

The idea can be formalized using the notation from Section 4.2. Let d_t denote the length of the shortest path between the source s and a sink $t \in T$ in the graph G. Without loss of generality, let t_1 be the sink that is nearest to s, and let $d_1 := d_{t_1} = \min_t d_t$. Since the source begins transmitting at the first time slot, the sinks start receiving on the d_1-st time slot at the earliest. Similarly, the source stops sending at the $(D - d_1 + 1)$-st time slot since all packets must arrive by the D-th time slot. Depending on the interference model used, it is now possible to calculate the exact time slot on which the source can start sending the next block of packets without mixing it with the current block.

For PI networks, one-hop neighbors of the source are no longer transmitting at the $(D - d_1 + 3)$-rd time slot. Thus, if the source transmits the next block of packets at the $(D - d_1 + 3)$-rd time slot, its one-hop neighbors can receive them without interference. This is equivalent to saying that the schedules G_1 and G_{D-d_1+3} are conflict free. Similar deductions can be made about G_2 and G_{D-d_1+4}, and so on. Define the union operation for two conflict-free schedules G_{ℓ_1} and G_{ℓ_2} as

$$G_{\ell_1} \cup G_{\ell_2} = (V, E_{\ell_1} \cup E_{\ell_2}). \tag{4.1}$$

The overall pipelined network schedule can thus be expressed as the sequence of graphs $G_1, G_2, \ldots, G_{D-d_1+3} \cup G_1, G_{D-d_1+4} \cup G_2, \ldots$. Further, the asymptotic throughput, given that Algorithm 4.1 returns μ edge-disjoint paths, becomes $\mu/(D - d_1 + 2)$.

The argument for SI networks is similar, except that the one-hop neighbors of the source can only receive when *their* next-hop neighbors stop transmitting. This happens at the $(D - d_1 + 4)$-th time slot, which yields an effective throughput of $\mu/(D - d_1 + 3)$.

In the analysis so far, it is assumed that in the worst case, the source may transmit the last packet to the sink t_1 at the $(D - d_1 + 1)$-st time slot. However, all other sinks are farther than d_1 hops and the number of packets reaching every sink is the same. Therefore, a more efficient choice of augmenting paths may make the source send its last packets at the $(D - d_2 + 1)$-st time slot where $d_2 = \max_t d_t$. This observation can be used to derive an upper bound on the achievable throughput. Specifically, if the source were to transmit one packet per-time slot, it is possible to transmit at most $D - d_2 + 1$ packets. If the augmenting paths are chosen carefully, transmission of the next block of packets may start immediately at time slot $D - d_2 + 2$, resulting in the maximum achievable throughput of 1. This is also the maximum achievable throughput for any JS-NC scheme, including those which do not consider delay constraints, since the source can only transmit at most one packet per-time slot.

The Earliest-Shortest Path (ESP) Algorithm

Recall that unlike the EK algorithm, it is possible to choose augmenting paths that may cause deletion of a large number of edges, thus yielding a small final value of μ. The choice of shortest augmenting paths is therefore a justifiable heuristic since shorter paths are expected to conflict with fewer edges. Another factor influencing the throughput returned by Algorithm 4.1 is the number of time slots for which each packet stays in the network. Intuitively, any packet that is transmitted within the first few time slots should be received by the sinks as soon as possible; or else, it may (unnecessarily) cause congestion to packets transmitted later. One way to ensure this is to always choose the shortest path $\mathcal{P}^{(t)}$ whose ending time slot (i.e., the time slot ℓ for which $t^r(\ell) \in \mathcal{P}^{(t)}$) is the least among all shortest paths.

This strategy can be implemented by using Dijkstra's algorithm to find the shortest path, but with a simple modification. Recall that Dijkstra's algorithm visits nodes starting at s^ν, and maintains an upper bound on the minimum distance from s^ν to each node. This upper bound is updated if a shorter distance is found, and the algorithm terminates when the entire graph has been visited. However, when all edges are of unit-length (as in the present case), if the nodes are visited in a breadth-first manner (i.e., the algorithm first visits all one-hop neighbors, then two-hop neighbors, and so on), the algorithm can terminate as soon as the destination is encountered.

The modified algorithm does not terminate on reaching the destination t^ν for the first time. Instead, a new variable $S_{\max}$ is initialized to store the time slot of the last visited receiver-subnode of t^ν. The next time t^ν is visited, the value of $S_{\max}$ is updated to the minimum of $S_{\max}$, and the time slot of the last-visited receiver-subnode. The algorithm terminates when all nodes as far from the source as t^ν have been visited. The earliest-shortest path can be recovered by backtracking along the receiver-subnode with time slot equal to the final value of $S_{\max}$. In other words, the ESP algorithm is similar to Dijkstra's, except that the time slot of the last visited receiver-subnode is used to break ties while choosing the shortest path at the sink node. The full ESP scheme is listed as Algorithm 4.2.

Algorithm 4.2: Earliest-shortest path (ESP) algorithm

```
1  Initialize $Q \leftarrow \{s^\nu\}$
2  Initialize variables $d_{\max} \leftarrow 0$, $d_{s^\nu} \leftarrow 0$, and $d_v \leftarrow \infty$ for all $v \in \mathcal{V} \setminus \{s^\nu\}$
3  Initialize $S_{\max} \leftarrow \infty$, and $S_v \leftarrow$ time slot associated with node $v$, for all
   $v \in \mathcal{V} \setminus \{s^\nu, \mathcal{T}\}$
4  repeat
5      $u \leftarrow \arg\min_{w \in Q} d_w$
6      $Q \leftarrow Q \setminus \{u\}$
7      $d_{\max} \leftarrow d_u$
8      foreach node $v$ in set $\{v|(u,v) \in \mathcal{E}\}$ do
9          if $v = t^\nu$ then
10             $S_{\max} \leftarrow \min(S_u, S_{\max})$
11         end
12         $d_v \leftarrow \min(d_v, d_u + 1)$
13         $Q \leftarrow \{Q \cup \{v\}\}$
14     end
15 until $d_{\max} = d_{t^\nu}$
16 Backtrack path starting from $t^r(S_{\max})$ to $s^\nu$.
```

Note that Algorithm 4.2 visits nodes in a breadth-first manner starting at s^ν. This is accomplished by maintaining a set Q of all nodes which have themselves been visited but whose neighbors have not been visited. At each iteration, the neighbors of a node closest to the source is visited and the distance metrics are updated. Interestingly, Algorithm 4.2 can be used to claim certain approximation guarantees for the PI model. This is established in the ensuing subsection.

4.3.2 Performance Bounds

Performance bounds are developed in this section for Algorithm 4.1 applied to PI networks. The following theorem gives a bound on the achievable throughput.

Theorem 4.1. *The throughput ρ obtained through Algorithm 4.1 using the ESP and pipelining enhancements can be bounded as follows:*

$$\frac{\left\lfloor \frac{D-d_2+2}{2} \right\rfloor}{D-d_1+2} \le \rho \le 1. \tag{4.2}$$

As a corollary, it can be seen that as $D \to \infty$, the bound reduces to $1/2 \le \rho \le 1$. Next, the proof of Theorem 4.1 is provided.

Proof. The upper bound has already been derived in Section 4.3.1. The proof of the lower bound relies on the special structure of the time-unwrapped graph $\mathcal{G}$. In particular, notice that the shortest $s^\nu - t^\nu$ path in $\mathcal{G}$ corresponds to the set of wireless nodes that lie on the shortest s-t path in G. Thus, for each sink t, there exist several shortest paths in $\mathcal{G}$ each of which has the following form

$$\begin{aligned}\mathcal{P}^{(t)}(\boldsymbol{\ell}) = \Big(&s^\nu, s^t(\ell_1), v_1^r(\ell_1), v_1^c(\ell_2), v_1^t(\ell_2),\\ &\ldots, v_{d_t-1}^r(\ell_{d_t-1}), v_{d_t-1}^c(\ell_{d_t}), v_{d_t-1}^t(\ell_{d_t}), t^r(\ell_{d_t}), t^\nu\Big)\end{aligned} \tag{4.3}$$

where $\boldsymbol{\ell} := (\ell_1, \ell_2, \ldots, \ell_{d_t})$ and $1 \le \ell_1 \le \ell_2 \le \ldots \le \ell_{d_t} \le D$. The length of the shortest $s^\nu - t^\nu$ path is therefore $3d_t + 1$. This is also the minimum length of the shortest augmenting path on any residual graphs that arise in Algorithm 4.1. In other words, the length of the shortest augmenting path always increases as iterations of Algorithm 4.1 go on. Such a behavior of increasing path-lengths is well known for the EK algorithm. It also holds here since any augmenting path found in Algorithm 4.1 is a feasible EK augmenting path.

For each sink $t \in T$, define the quickest-shortest (QS) path starting at time slot ℓ as $Q^{(t)}(\ell) := \mathcal{P}^{(t)}(\boldsymbol{\ell})$, where $\boldsymbol{\ell} = (\ell, \ell+1, \ell+2, \ldots, \ell+d_t-1)$. Note that given any graph $\mathcal{G}$, such a path exists for every sink $t \in T$ and every time slot $1 \le \ell \le D - d_t + 1$. Further, starting at time slot ℓ, $Q^{(t)}(\ell)$ is a shortest path that reaches the sink $t^\nu \in \mathcal{T}$ at the earliest possible time slot $\ell + d_t - 1$. Thus, for a given sink $t \in T$ Algorithm 4.2 will return the QS path $Q^{(t)}(\ell)$ for some ℓ, as long as such a path exists in the residual graph. Note that the QS paths $\{Q^{(t)}(\ell)\}_{t\in T}$ do not conflict with each other, and thus form a shortest path tree (SPT).

Next, define a partial order on all shortest augmenting paths returned by Algorithm 4.2 with starting time slot ℓ_1 and ending time slot ℓ_{d_t}. Specifically, given two augmenting paths $\mathcal{P}^{(t_1)}(\boldsymbol{i})$

and $\mathcal{P}^{(t_2)}(\boldsymbol{j})$, ending at two, possibly different sinks t_1 and t_2, define $\mathcal{P}^{(t_1)}(\boldsymbol{i}) \leq \mathcal{P}^{(t_2)}(\boldsymbol{j})$ if and only if $i_1 \leq j_1$ *and* $i_{d_{t_1}} \leq j_{d_{t_2}}$. The partial ordering can now be used to understand Algorithm 4.2 better. For instance, if at some iteration in Algorithm 4.1, it is known that a path $\mathcal{P}_1^{(t)}$ exists in the residual graph, then Algorithm 4.2 will always return a path $\mathcal{P}_2^{(t)} \leq \mathcal{P}_1^{(t)}$ of length less than or equal to the length of $\mathcal{P}_1^{(t)}$.

The following lemma states another useful aspect of the QS paths.

Lemma 4.1. *If t_a and t_b be two (possibly different) sinks, and ℓ denotes any time slot such that the paths $Q^{(t_a)}(\ell)$ and $Q^{(t_b)}(\ell+2)$ exist on the graph $\mathcal{G}$, then the path $Q^{(t_b)}(\ell+2)$ does not conflict with any path $\mathcal{P} \leq Q^{(t_a)}(\ell)$.*

The proof of Lemma 4.1 is provided in Appendix 4.A. The result is interesting in the sense that the entire SPT formed by QS paths starting at a given time slot ℓ does not conflict with any path in the SPT starting at time slot $\ell + 2$. This observation can now be used to prove the main result of Theorem 4.1 as follows.

(L1) In the first iteration, the paths $Q^{(t)}(1)$ for each sink $t \in T$ exist and are therefore returned by Algorithm 4.2. At the end of the first iteration, all edges that lie on any of the paths $Q^{(t)}(1)$ are reversed and assigned unit flow values.

(L2) At the second iteration, any ESP starts at or after the second time slot. While the paths $Q^{(t)}(2)$ may not necessarily exist, Lemma 4.1 ensures that the QS paths $Q^{(t)}(3)$ still exist; that is, they are not deleted from the residual graph in the first iteration. As observed earlier, an ESP returned at the second iteration is such that $\mathcal{P}^{(t)} \leq Q^{(t)}(3)$.

(L3) Generically, the i-th iteration returns a path $\mathcal{P}^{(t)} \leq Q^{(t)}(2i+1)$. Since the farthest sink allows the source to transmit up to the $(D - d_2 + 1)$-th time slot, there are at least $\left\lfloor \frac{D-d_2+2}{2} \right\rfloor$ iterations, and as many augmenting paths.

(L4) As observed earlier, transmission of the next block can begin at time slot $(D - d_1 + 3)$. This yields the asymptotic throughput of $\frac{\lfloor (D-d_2+2)/2 \rfloor}{D-d_1+2}$.

$\square$

Note that for the SI model, it is not possible to provide similar guarantees as the QS paths for different sinks $t \in T$ starting at the same time slot $Q^{(t)}(\ell)$ may conflict with each other. Thus, the existence of the SPT itself is not guaranteed. However, the proof of Theorem 4.1 provides some justification for the ESP heuristic even when the algorithm is applied to generic interference models.

4.3.3 Distributed Implementation

Algorithm 4.1 readily lends itself to a distributed implementation. Assume that each wireless node is aware of its two-hop neighbors, the source and sink nodes, and the graph parameters D and d_t for each $t \in T$. The following observations may then be used to distribute the algorithm.

(D1) Construction of the time-unwrapped graph $\mathcal{G}$ only involves creation of several subnodes per node, which can be done locally, without requiring any communication among the nodes.

(D2) The source must calculate the ESP for every iteration and every sink. Distributed and asynchronous versions of Dijkstra's algorithm are available [121, Chap. 5], and can be readily adapted for Algorithm 4.2 here. A speed improvement can be obtained by always visiting the nodes with an earlier time slot first.

(D3) Finally, the source sends a packet along the shortest augmenting path, found in D2, informing the nodes of its choice. The participating nodes may then obtain the residual graph, update flow values along their edges, and delete conflicting edges by informing their neighbors.

Before ending this section, a few remarks are in order.

Remark 4.5. During the operation, each node in Algorithm 4.1 transmits and receives on predetermined time slots with a fixed schedule. This allows most nodes to sleep for most of the time slots, except when operating or performing maintenance tasks. This aspect of Algorithm 4.1 makes it attractive for sensor and ad hoc networks.

Remark 4.6. Most deterministic network code designs, such as those in [78], result in relatively small finite field sizes, typically $O(|T|)$. Randomized schemes such as the one in [73] only require a field size q that is a prime power greater than $|T|$. This is in contrast with most random network coding schemes that assume asymptotically large field sizes (usually 2^8 or 2^{16}). Smaller field sizes translate to lower overhead since the coding coefficients are usually carried in the packet headers [27].

Remark 4.7. The distributed version of the algorithm works in a feed-forward way. Thus, neither link-by-link nor end-to-end acknowledgments are required. Such an ACK-free operation makes sense in networks with hard deadlines since nodes do not have time for re-transmissions anyway. This is appealing for video streaming applications, where feed-forward operation is commonly used; see e.g. [137].

4.4 Linear Programming Bounds

This section examines the maximization of μ from an integer programming perspective. Section 4.4.1 describes an integer programming formulation for PI networks. While it may be impossible to efficiently solve the resultant integer program for large networks, the formulation provides ways of obtaining upper bounds. For example, a linear programming (LP) bound is obtained in Section 4.4.1 by relaxing the integrality constraints in the integer program. Section 4.4.2 further improves this bound by adding a class of *valid inequalities*.

4.4.1 Integer Programming Formulation

Following the notation of Section 4.2, the problem of finding the maximum number of edge-disjoint paths from the source s^ν to each of the sinks $t^\nu \in \mathcal{T}$, can be expressed as the following integer program:

$$\mu^* = \arg\max \mu \tag{4.4a}$$

$$\text{s. t.} \quad \sum_{e\in I_v} x_e^{(t)} = \sum_{e\in O_v} x_e^{(t)}, \qquad t \in T, v \in \mathcal{G} \setminus (s^\nu, t^\nu) \tag{4.4b}$$

$$\sum_{e\in O_s} x_e^{(t)} = \mu, \qquad t \in T \tag{4.4c}$$

$$\sum_{e\in I_t} x_e^{(t)} = \mu, \qquad t \in T \tag{4.4d}$$

$$z_e \geq x_e^{(t)}, \qquad t \in T, e \in \mathcal{E} \tag{4.4e}$$

$$x_e^{(t)}, z_e \in \{0,1\}, \qquad t \in T, e \in \mathcal{E} \tag{4.4f}$$

where variables $x_e^{(t)}$ and z_e represent the virtual and real flows, respectively, on the edge $e \in \mathcal{E}$ [97]. The flow variables are related to the flows defined in Section 4.3. The virtual flow $x_e^{(t)}$ corresponds to the flow values assigned to edges on $\mathcal{G}_t$, while the real flow z_e corresponds to the overall flow on $\mathcal{G}$.

In the wireless setting, the scheduling constraints SC1-SC2(a) must also be added. For a node v and time slot k, these constraints can be represented by the inequality

$$\sum_{e\in I_{v^r(k)}} z_e + z_{(v^c(k),v^t(k))} \leq 1 \qquad \forall v \in \mathcal{V}, 1 \leq k \leq D \tag{4.4g}$$

where the first summand in (4.4g) represents the total flow on edges incoming to the receiver-subnode $v^r(k)$, and the second term is the flow leaving the combiner/transmitter-subnode at the same time slot. The inequality ensures that in a single time slot k, at most a single packet is either received (from a single node) or transmitted (broadcast to possibly multiple nodes).

The LP bound for the problem (4.4a)–(4.4g) can be obtained by relaxing (4.4f) with,

$$x_e^{(t)}, z_e \in [0,1]. \tag{4.4h}$$

This bound can be further improved by adding "tightening" inequalities that are valid only for the original integer programming constraints. In other words, these valid inequalities may "cut-off" regions of the polyhedron defined by the linear inequality constraints (4.4b)–(4.4e) and (4.4g)–(4.4h). Valid inequalities can also be used to exactly solve the integer program using methods such as branch-and-cut [167, Chap. 8], although the worst-case complexity of these integer programming solvers is still not polynomial. The next subsection focuses on developing a set of such valid inequalities.

4.4.2 A Class of Valid Inequalities

Before describing the valid inequalities, some simplifications and related notation is introduced. First note that it is straightforward to eliminate the variable μ from the set of equations (4.4b)–(4.4d). Next, let $\mathbf{w}$ be the $n \times 1$ super-vector that contains all remaining optimization variables $\{x_e^{(t)}, z_e\}$. After eliminating μ, the constraints (4.4b)–(4.4e) and (4.4g) can be generically denoted by the set of inequalities $\mathbf{Aw} \leq \mathbf{b}$, where each equality constraint is simply expressed as two opposing inequalities. The set of all feasible integer programming solutions is then given by $\{\mathbf{w} \in \{0,1\}^n | \mathbf{Aw} \leq \mathbf{b}\}$, while the corresponding LP relaxation lies in the polyhedron represented as the set $\{\mathbf{w} \in [0,1]^n | \mathbf{Aw} \leq \mathbf{b}\}$. A set of inequalities $\mathbf{Cw} \leq \mathbf{d}$ is said to be valid if

$$\{\mathbf{w} \in \{0,1\}^n | \mathbf{Aw} \leq \mathbf{b}\} = \{\mathbf{w} \in \{0,1\}^n | \mathbf{Aw} \leq \mathbf{b}, \mathbf{Cw} \leq \mathbf{d}\} \tag{4.5}$$

while

$$\{\mathbf{w} \in [0,1]^n | \mathbf{Aw} \leq \mathbf{b}\} \supseteq \{\mathbf{w} \in [0,1]^n | \mathbf{Aw} \leq \mathbf{b}, \mathbf{Cw} \leq \mathbf{d}\}. \tag{4.6}$$

It is well known that the optimum solution of an LP always lies on an extreme point of the polyhedron defined by its linear inequalities [167, Chap. 2]. For an integer program however, its LP-relaxation polyhedron may not necessarily have integral extreme points. The optimum solution of the LP may therefore be fractional, and its optimum value may lie far from the optimum of the integer program. Valid inequalities can be used in such cases to cut-off some or all of the fractional extreme points of the LP relaxation polyhedron.

In principle, a finite number of "necessary" valid inequalities is sufficient to ensure that all extreme points of $\{\mathbf{w} \in [0,1]^n | \mathbf{Aw} \leq \mathbf{b}, \mathbf{Cw} \leq \mathbf{d}\}$ are integral. A well-known method of generating valid inequalities is the Chvatal-Gomory (CG) procedure, which can generate all necessary valid inequalities in a finite number of steps [167, Chap. 8]. Given a system of m linear inequalities, $\mathbf{Aw} \leq \mathbf{b}$, and a vector $\mathbf{g} \in [0,1)^m$, the CG procedure generates the valid

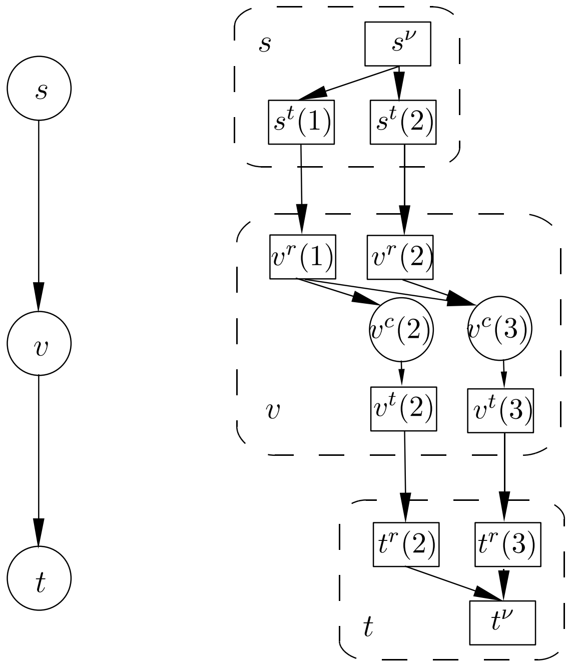


Figure 4.3: An example of wireless network and its time-expanded version.

inequality (also called a CG-cut) denoted as

$$\lfloor \mathbf{g}^T \mathbf{A} \rfloor \mathbf{w} \leq \lfloor \mathbf{g}^T \mathbf{b} \rfloor \tag{4.7}$$

where $\lfloor \boldsymbol{\alpha} \rfloor$ stands for the elementwise floor operation of the vector $\boldsymbol{\alpha}$.

The caveat however is that the CG-procedure generates an exponentially large set of inequalities, which cannot be handled efficiently by any LP solver. The remainder of this section describes a method to generate a smaller class of valid inequalities that can be efficiently *separated*, and thus accommodated by LP solvers.

Figure 4.3 depicts a simple example network and its time-expanded graph for $D = 3$, with redundant nodes and edges removed. The solution obtained for this network using the LP relaxation is, $x_e = z_e = 1$ for $e = (s^\nu, s^t(1)), (s^t(1), v^r(1)), (v^t(3), t^r(3)), (t^r(3), t^\nu)$, and $x_e = z_e = 0.5$ for all other $e \in \mathcal{E}$. This gives a total flow of 1.5 that also satisfies the constraints (4.4g). The integral solution, on the other hand, achieves only one unit of end-to-end flow, i.e., a single packet is transmitted from s to v in the first time slot, and from v to t in the second time slot.

An observation that follows from this example is that in three time slots, only one packet goes "through" v. Interestingly, this also holds for larger values of D and for any three, possibly non-contiguous, time slots. For instance there can be at most three packets, each either transmitted or received by a node in three time slots k_1, k_2 and k_3. However, there can be at most one packet which is both transmitted *and* received in these three time slots. In contrast, the constraint (4.4g) allows 1.5 packets to be transmitted and received.

In order to enforce this condition, note that the flow passing through a node v in time slots k_1, k_2 and k_3, is given by the total flow through the edges in $\mathcal{C} := \{(n_1, n_2) | n_1 \in \{v^r(k_1), v^r(k_2)\}, n_2 \in \{v^c(k_2), v^c(k_3)\}\}$. Thus, an extra inequality can be introduced, limiting the total flow on these edges to one. Note that for the case of multiple sinks, the virtual flow corresponding to each sink requires a separate inequality. The idea can be generalized to any odd number of time slots as asserted by the following theorem.

Theorem 4.2. *For the time slots $1 \le k_1, k_2, \ldots, k_\ell \le D$, and a node v, define the set of edges,*

$$\begin{aligned}\mathcal{C} := \{(n_1, n_2) | n_1 \in \{v^r(k_1), v^r(k_2), \ldots, v^r(k_{\ell-1})\},\\ n_2 \in \{v^c(k_2), v^c(k_3), \ldots, v^c(k_\ell)\}\}.\end{aligned} \tag{4.8}$$

Then, the following is a valid inequality for all $t \in T$

$$\sum_{e \in \mathcal{C}} x_e^{(t)} \le \left\lfloor \frac{\ell}{2} \right\rfloor . \tag{4.9}$$

Proof. The cuts can be generated by applying $\{0, \frac{1}{2}\}$-CG cuts to some of the constraints in (4.4b)–(4.4e) and (4.4g). Note that for any slot k, the following holds

$$\begin{aligned} & \sum_{e \in I_{v^r(k)}} z_e + z_{(v^c(k), v^t(k))} \le 1 \\ \overset{\text{[cf. (4.4e)]}}{\Rightarrow} & \sum_{e \in I_{v^r(k)}} x_e^{(t)} + x_{(v^c(k), v^t(k))}^{(t)} \le 1 \\ \overset{\text{[cf. (4.4b)]}}{\Rightarrow} & \sum_{e \in O_{v^r(k)}} x_e^{(t)} + \sum_{e \in I_{v^c(k)}} x_e^{(t)} \le 1 \end{aligned} \tag{4.10}$$

Adding up the last set of equations for slots $k = k_1, k_2, \ldots, k_\ell$, we obtain

$$\sum_{k \in \{k_1, \ldots, k_\ell\}} \sum_{e \in O_{v^r(k)}} x_e^{(t)} + \sum_{k \in \{k_1, \ldots, k_\ell\}} \sum_{e \in I_{v^c(k)}} x_e^{(t)} \le \ell. \tag{4.11}$$

Note that in (4.11), the terms $x_e^{(t)}$ for all $e \in \mathcal{C}$ occur twice, while all other terms occur only once. Thus, dividing (4.11) by 2 and rounding towards zero, we arrive at (4.9). □

The generated set of inequalities (4.9) is much smaller than the full set of all possible valid inequalities, and is therefore not necessarily optimal. Further, even this set contains exponentially many constraints. Interestingly however, the set admits an efficient separation oracle which identifies a possibly violated inequality given any feasible solution of the relaxed LP problem. The solution to the entire LP can also be found efficiently using the ellipsoid method [148, Chap. 5], whose calls to the separation oracle can be bounded polynomially. For the set of inequalities generated by (4.9), the following result holds.

Lemma 4.2. *The worst-case complexity of the separation oracle is $O(nD|T|)$.*

Proof. Given a candidate solution $(x_e^{(t)}, z_e)$, the problem of verifying its feasibility can be stated as follows:

For every node v and sink t,

(F1) Find the set of time slots $k_1, \ldots, k_\ell$ such that a constraint in (4.9) is violated; or,

(F2) Output that there is no such set.

It will be argued next that this problem is equivalent to finding a separation oracle for the *matching* problem on a derived graph. Given a graph $G_v = (U_v, E_v)$, associate variables y_e^v with each edge $e \in E_v$. Let C_u denote the set of edges connected to a node $u \in U_v$. A matching is a set of edges, such that no two edges of the set connect to the same node. Equivalently, a matching is an assignment of binary values to variables y_e^v such that

$$\sum_{e \in C_u} y_e^v \leq 1, \qquad\qquad y_e^v \in \{0,1\}. \tag{4.12a}$$

Interestingly, it is possible to replace the integrality constraints in (4.12a) with simple non-negativity constraints, by adding the following set of valid inequalities

$$\sum_{\{e=(u_1,u_2) | u_1,u_2 \in \mathcal{S}\}} y_e^v \leq \left\lfloor \frac{|\mathcal{S}|}{2} \right\rfloor \ \forall \text{ sets of nodes } \mathcal{S}. \tag{4.12b}$$

Although the number of valid inequalities in (4.12b) is also exponential, it is possible to design a separation oracle that returns the violated inequality in $O(|U_v|)$ [148, Chap. 25].

In the present case, for a node v, construct graph G_v with nodes 1, 2, …, D, and connect pairs of nodes (i, j) for all $i > j$. The edge (i, j) in G_v represents the edge $(v^r(i), v^c(j))$ in the original graph $\mathcal{G}$. Similarly, set the variables y_{ij}^v equal to the corresponding edge variables $x_{(v^r(i),v^c(j))}^{(t)}$. A related set of constraints for y_{ij}^v can be derived based on (4.4g) and (4.9) as follows.

(M1) All flow variables are positive, thus implying $y_e^v \geq 0$ for all $e = (i, j)$.

(M2) The set of edges connecting to a node $k \in \{1, \ldots, D\}$ in G_v correspond to all the edges $e \in O_{v^r(k)} \cup I_{v^c(k)}$. Thus, (4.10) implies that

$$\sum_{e \in C_k} y_e^v \leq 1. \tag{4.13a}$$

(M3) Since any set of time slots $\{k_1, \ldots, k_\ell\}$ corresponds to an equivalent set of nodes in G_v, (4.9) translates to

$$\sum_{\{e=(k_i,k_j)|1\leq i,j\leq \ell\}} y_e^v \leq \left\lfloor \frac{\ell}{2} \right\rfloor \qquad \forall\, 1 \leq \ell \leq D. \tag{4.13b}$$

It can be seen that the constraints (4.13a)–(4.13b) resemble the matching constraints (4.12a)–(4.12b). Thus, given a candidate solution $x_e^{(t)}$, an assignment to variables y_e^v can be calculated for each node v. Invoking the matching separation oracle then results in a possibly violated inequality in terms of y_e^v, which can finally be translated to a corresponding inequality in terms of $x_e^{(t)}$. Since the separation oracle runs in time $O(D)$ and must be invoked for every node and every virtual flow, the total time complexity is $O(nD|T|)$. $\square$

4.5 Numerical Comparisons

This section presents simulations on the performance of Algorithm 4.1. For comparison, the throughput obtained using a delay-agnostic, conflict-graph method from [71] along with the bounds derived in Sections 4.3.2 and 4.4, are also plotted.

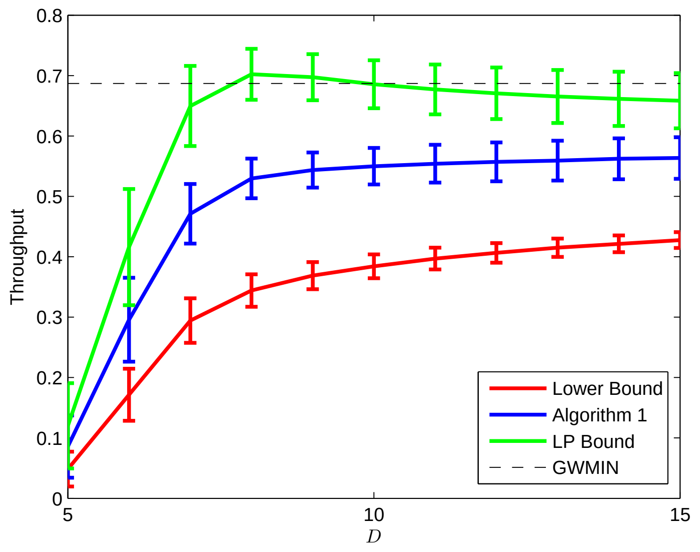


Figure 4.4: Performance and bounds on a PI network.

Random networks are generated using the MAX-DPA algorithm outlined in [117]. The algorithm generates graphs by placing nodes one-by-one, while respecting certain maximum-degree and proximity constraints so as to simulate a realistic ad hoc network. The algorithm parameters are chosen to be $d = 5$, $d_{\max} = 8$, and $d_0 = 0.2$, which denote respectively the average and maximum node degrees, and the minimum distance between neighbors. The nodes are placed in a square area chosen such that the average node density is one. Next, the leftmost node is chosen to be the source and all edges are chosen to be directed away from the source. Finally, all nodes without any outgoing edges are chosen to be sinks.

Figure 4.4 shows the performance of Algorithm 4.1 for small PI networks. The throughput is averaged over 2,000 different networks with 20 nodes each, and is plotted for a range of values of the deadline D. The length of the bars equals half the standard deviation over the network instances. Pipelining in Algorithm 4.1 is implemented such that the source does not necessarily wait till the $(D-d_1+3)$-rd time slot, but may begin transmission earlier if possible. For comparison, the lower bound stated in Theorem 4.1 and the LP upper bounds are also plotted. In the absence of any deadlines, it is possible to evaluate the maximum achievable throughput using one of the approximation algorithms outlined in [71]. The dashed line in

D	15	20	25	30	35	Random approach [71]
Network 1	0.33	0.36	0.37	0.38	0.38	0.50
Network 2	0.20	0.20	0.24	0.25	0.27	0.39
Network 3	0.25	0.21	0.29	0.30	0.28	0.39

Table 4.1: Performance of Algorithm 4.1 on large networks

the figure shows this value, calculated using the random approach of [71] with 500 random maximal-independent sets, assuming no erasures on links.

As expected, Algorithm 4.1 exhibits graceful degradation in performance as the deadline is reduced. Interestingly, the trend is also visible in the curves showing upper and lower bounds on the throughput. Further, it can be seen that the bounds become tighter as the value of D increases, reaffirming their usefulness. Finally, note that the variation apparent from the standard deviation bars is largely because of the variation among random networks. Thus, the overlap between the bars for lower and upper bound does *not* mean that the bounds are incorrect for some network instances.

Next, the performance is analyzed on a large network with 100 nodes and SI constraints. Three random networks are generated, and Algorithm 4.1 is run for different values of D. While the lower and upper bounds do not apply to this case, the random approach from [71] is again used here as a benchmark. Note that for large networks, the number of hyperarcs, and consequently the size of the resulting LP used in [71] becomes prohibitive. Towards this end, the interference model used in [71] is simplified by considering only broadcast transmissions, i.e., each node either broadcasts its packets to all its receivers or stays silent. This translates to the simple rule, used in several MAC protocols: a node transmits only when its two-hop neighborhood is silent. Further, only 200 maximal-independent sets are generated. Table 4.1 lists the throughput achieved for all three realizations. As with the PI model, the throughput-delay trade off is again apparent here. In this case however, the difference between the deadline-free case and the GAP throughput with large D is not as pronounced. Further, the quality of approximation in Algorithm 4.1 also depends on the topology of the network. Thus for some networks, such as Network 3 in Table 4.1, the throughput does not always decrease monotonically with

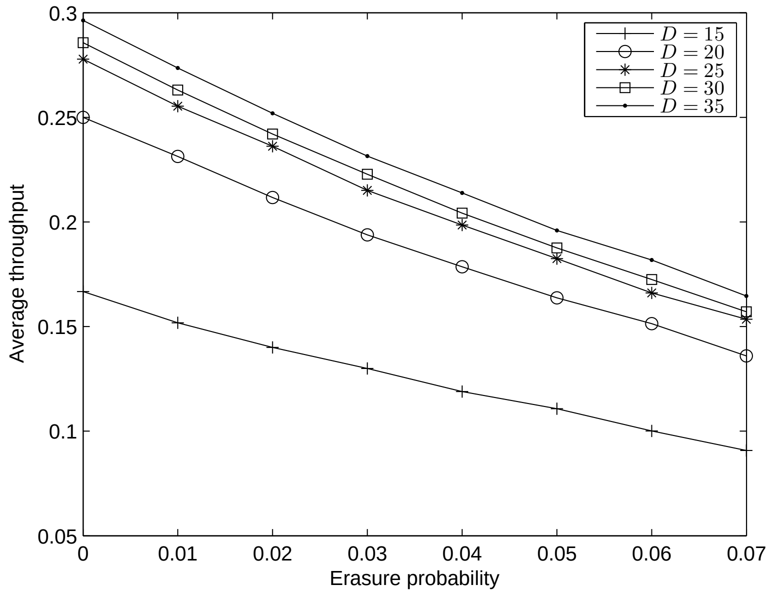


Figure 4.5: Degradation of throughput with packet erasures for different values of D.

D.

Finally, the performance of the resulting network protocol is studied for different erasure probabilities. Towards this end, a random network with 100 nodes is generated, and Algorithm 4.1 is run to obtain the network operation schedules. Next, the protocol is simulated using Monte-Carlo runs, assuming that the links fail independently with specified erasure probabilities. Figure 4.5 depicts the average throughput, given by the average number of linear combinations received by the sinks, per time slot. It can be seen that the throughput performance degrades only gradually with erasures.

4.6 Conclusion

This chapter considered network-coded multicast with deadline constraints. Since popular generation-based approaches do not handle delay constraints, a joint scheduling and network coding approach is introduced to maximize the average throughput while respecting the wireless constraints and packet-deadlines. The novel algorithm relies on a time-unwrapped graph expansion in order to construct linear-periodic time-varying network codes. The approach draws from the well-known augmenting-path algorithm, and is therefore both distributed and scalable. For networks with primary interference constraints, the algorithm was shown to have a constant-factor bounded worst-case performance. The setup was also analyzed from an integer programming perspective, and a set of valid inequalities was developed and used to obtain a linear programming based upper bound on the throughput.

4.A Proof of Lemma 4.1

First, using contradiction, we show that paths $Q^{(t_a)}(\ell)$ and $Q^{(t_b)}(\ell+2)$ do not conflict. If the two said paths indeed conflict, it would imply that there exist $e_a \in Q^{(t_a)}(\ell)$ and $e_b \in Q^{(t_b)}(\ell+2)$ such that one of the following holds:

(C1) Edges e_a and e_b violate the half-duplex constraint. This means that there exists a node $v \in V$ and a time slot $\ell \leq k \leq \min(d_{t_a}, d_{t_b})$ such that either (a) $e_b \in I_{v^r(k)}$ and $e_a \in O_{v^t(k)}$; or, (b) $e_a \in I_{v^r(k)}$ and $e_b \in O_{v^t(k)}$.

(C2) There exists a node $v \in V$ and time slot k such that $e_a \in I_{v^r(k)}$ and $e_b \in I_{v^r(k)}$.

(C3) The two edges are the same, i.e., $e_a = e_b$.

We begin by assuming that (C1-a) holds for some time slot k and node v. Since node $v^r(k+1)$ lies on the QS path $Q^{(t_a)}(\ell+2)$, it implies that a subnode in the wireless node v can be reached in $k-\ell-1$ time slots if the path along $Q^{(t_a)}(\ell+2)$ is taken. Note however that $v^r(k)$ also lies on $Q^{(t_b)}(\ell)$, which would imply that it must take at least $k-\ell$ time slots if a path along $Q^{(t_b)}(\ell)$ is taken. Therefore, the path $Q^{(t_a)}(\ell)$ reaches node v earlier than the path $Q^{(t_b)}(\ell)$ starting at the same slot. This is a contradiction since both paths were already assumed to be QS paths. The intuition is that starting at time slots ℓ and $\ell+2$, the time slots at which two QS paths reach a node differ by at least two.

The complementary case (C1-b) yields an even stronger contradiction as it implies that the QS path starting at a later time slot reaches a node at an earlier one. Similarly, the other cases C2 and C3 also follow from the aforementioned argument. Specifically, both C2 and C3 imply that two QS paths, starting at different time slots ℓ and $\ell+2$, reach a node at the same time slot k, which is not possible. It can be seen that the argument holds if the path $Q^{(t_a)}(\ell)$ is replaced by a shortest path $\mathcal{P} \leq Q^{(t_a)}(\ell)$ since that would again imply a stronger contradiction.

Note that it is not possible to provide similar guarantees for the SI model since, unlike the PI model, two QS paths starting at the same time slot may conflict with each other.

Chapter 5

Network-Compressive Coding for Wireless Sensors Networks

A network-compressive transmission protocol is developed in which correlated sensor observations belonging to a finite alphabet are linearly combined as they traverse the network on their way to a sink node. Statistical dependencies are modeled using factor graphs. The sum-product algorithm is run under different modeling assumptions to estimate the maximum a posteriori set of observations given the compressed measurements at the sink node. Error exponents are derived for cyclic and acyclic factor graphs using the method of types, showing that observations can be recovered with arbitrarily low probability of error as the network size grows. Simulated tests corroborate the theoretical claims.

This chapter is organized as follows. Section 5.1 describes the model, and Section 5.2 describes the sum-product variants for cyclic and acyclic factor graphs. Section 5.3 derives the error exponents when exact MAP decoding is possible. Section 5.4 gives simulation results with synthetic and real datasets for both cyclic and acyclic cases. Finally, Section 5.5 concludes the chapter.

5.1 System Model and Problem Formulation

Consider a sensor network with a set of nodes $\mathcal{N}$, deployed to observe an environmental phenomenon. The environmental state, at the location of a sensor $n \in \mathcal{N}$, is represented by a discrete random variable Θ_n, taking values $\theta_n \in \mathbb{F}_Q$, where $\mathbb{F}_Q$ denotes the finite field of alphabet size Q. The state variables are assumed drawn from a known prior probability mass function (pmf) $p(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ stacks the variables $\{\theta_n\}$. Sensor $n \in \mathcal{N}$ does not directly observe θ_n, but instead its noisy version $x_n \in \mathbb{F}_Q$, drawn independently from a known pmf $p(x_n|\theta_n)$. Next, the $N := |\mathcal{N}|$ sensor observations, henceforth denoted by the $N \times 1$ vector $\mathbf{x}$, are communicated to a sink node (fusion center) t. Linear network coding is used to combine entries in $\mathbf{x}$ as they traverse the network on their way to the sink, which receives the $M \times 1$ vector $\mathbf{y} = \mathbf{A}\mathbf{x}$, where entries of $\mathbf{A}$ are also drawn from $\mathbb{F}_Q$ and are known at the sink node. Given $p(\boldsymbol{\theta})$, $p(x_n|\theta_n)$, $\mathbf{A}$, and $\mathbf{y}$, the sink wishes to estimate $\boldsymbol{\theta}$.

In order to motivate the system model, consider sensor networks deployed in tracking applications, where the environmental state takes only two possible values, corresponding to the presence or absence of a target. Moreover, since only a few sensors may detect the target at a given instant, the state variables are clearly correlated among nearby sensors. The observation noise is also binary in this case, arising from false positives or false negatives in the detectors of the individual sensor nodes.

In environment-monitoring systems, while many natural phenomena are continuous-valued, one may be interested only in monitoring them coarsely. For instance in monitoring levels of a chemical contaminant or temperature, the quantity of interest may only be the quantized value, say in whole degrees centigrade. In this case, the environmental state can be modeled again as a discrete random variable, representing the quantized version of the analog-amplitude quantity. Moreover, since continuous values at nearby sensor nodes are correlated, their quantized components will also be correlated. Finally, observations, which are the quantized and noisy versions of the true analog-amplitude quantity, can be modeled as the noisy version of the quantized values.

Given $\mathbf{y}$ at the sink node, the *a posteriori* probability is given by

$$\begin{aligned}
p(\boldsymbol{\theta}|\mathbf{y}) &\propto p(\boldsymbol{\theta}, \mathbf{y}) && \text{(5.1a)}\\
&= \sum_{\mathbf{x}\in\mathbb{F}_Q^N} p(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) && \text{(5.1b)}\\
&= \sum_{\mathbf{x}\in\mathbb{F}_Q^N} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}) && \text{(5.1c)}\\
&= \sum_{\mathbf{x}\in\mathbb{F}_Q^N} p(\mathbf{y}|\mathbf{x}) \prod_{n=1}^{N} p(x_n|\theta_n)p(\boldsymbol{\theta}) && \text{(5.1d)}
\end{aligned}$$

where (5.1c) follows from the fact that $\mathbf{x}(t)$ and $\mathbf{y}$ are conditionally independent given $\mathbf{x}$. Here $p(\mathbf{y}|\mathbf{x})$ is simply the indicator function $\mathbb{1}_{\mathbf{y}=\mathbf{A}\mathbf{x}}$, and (5.1d) follows independence assumption on the observation noise, which implies $p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(x_n|\theta_n)$. The sink node wishes to obtain the block maximum a posteriori (MAP) estimate of $\boldsymbol{\theta}$, that is

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\mathbb{F}_Q^N} p(\boldsymbol{\theta}|\mathbf{y}). \tag{5.2}$$

Alternatively, the sink seeks the a posteriori probability (APP) of each θ_n, namely

$$p_n(\theta_n|\mathbf{y}) = \sum_{\boldsymbol{\theta}\in\mathbb{F}_Q^N\backslash\theta_n} p(\boldsymbol{\theta}|\mathbf{y}) \tag{5.3}$$

where the notation $\backslash\theta_n$ is used to indicate that the sum is carried over all $\boldsymbol{\theta} \in \mathbb{F}_Q^N$ with fixed θ_n. From (5.3), the per-entry MAP estimate can be found as $\hat{\theta}_n = \arg\max_{\theta_n\in\mathbb{F}_Q} p_n(\theta_n|\mathbf{y})$.

In general, finding (5.2) or (5.3) involves searching or summing over an exponentially large space. Similar problems involving maximization (or marginalization) of a pmf over a discrete domain are encountered in several areas, most notably in channel decoding, image processing, and statistical physics [100]. To cope with this prohibitive complexity, factor graph representations of $p(\boldsymbol{\theta}|\mathbf{y})$ are often used to perform such maximization (or marginalization) at least approximately. In the present chapter, the sum-product algorithm is employed to efficiently evaluate the per-entry MAP. The sum-product algorithm has also been proposed for a related problem considered in [101]. In general however, the performance of message-passing algorithms may not necessarily be reliable, and the sum-product algorithm may not even converge.

The focus here is therefore on identifying scenarios where the prior pmf $p(\boldsymbol{\theta})$ and the coding matrix $\mathbf{A}$ have enough structure so as to guarantee convergence and asymptotic optimality.

Note that unlike traditional network coding schemes, matrix $\mathbf{A}$ need not be square since the correlation of $\mathbf{x}$ and $\boldsymbol{\theta}$ can be utilized to solve (5.2) or (5.3) even when $M < N$. Clearly in this case compression is achieved, with ratio $\eta = M/N$. Before concluding this section, a remark about the practical implementation aspects of the algorithm is due.

Remark 5.1. In low-cost sensor networks, MAC protocols (such as S-MAC [174]) often use packetized transmissions instead of transmitting individual observations. A packet may aggregate multiple observations collected over time, of the same or multiple physical quantities. Packetization is achieved in the proposed algorithm by assigning $\log_2 Q$ bits per observation in each packet. The entries of the $\mathbf{A}$ matrix are chosen by the intermediate nodes, and the same linear combination is used for all observations within a packet. These entries are then stored in the packet headers, so that they can be used by the sink for decoding without significant overhead; see e.g. [27, 76]. Finally, note that packets may be lost due to communication errors, and this may result in the sink receiving fewer than M linear combinations. The proposed algorithm is still applicable for this case, since the matrix $\mathbf{A}$, constructed from the received packet headers, will only contain rows corresponding to the correctly received y_m. Sensor failures can also be handled similarly, by setting the entries of the corresponding column in $\mathbf{A}$ to zeros.

5.2 Factor Graph Representation and Message-Passing Algorithm

The per-sensor posterior probability $p_n(\theta_n|\mathbf{y})$ can be expressed as

$$p_n(\theta_n|\mathbf{y}) = \sum_{\boldsymbol{\theta}\in\mathbb{F}_Q^N\setminus\theta_n} \sum_{\mathbf{x}\in\mathbb{F}_Q^N} p(\boldsymbol{\theta},\mathbf{x}|\mathbf{y}) \tag{5.4}$$

$$\propto \sum_{\boldsymbol{\theta}\in\mathbb{F}_Q^N\setminus\theta_n} \sum_{\mathbf{x}\in\mathbb{F}_Q^N} p(\mathbf{y}|\mathbf{x}) \prod_{n=1}^{N} p(x_n|\theta_n) p(\boldsymbol{\theta}). \tag{5.5}$$

Efficient evaluation of the summation in (5.5) may be possible if the multiplicands can be further factored into several terms, each depending on only a subset of variables in $\boldsymbol{\theta}$, $\mathbf{x}$, and $\mathbf{y}$. Towards this end, the following modeling assumptions are made.

(A1) The pmf $p(\boldsymbol{\theta})$, describing the hidden random variables, can be factored as

$$p(\boldsymbol{\theta}) = \frac{1}{Z} \prod_{j=1}^{J} f_{C_j}(\boldsymbol{\theta}_{C_j}) \tag{5.6}$$

where $C_1, \ldots, C_J \subset \mathcal{N}$ are generally overlapping clusters (or *cliques*) of nodes, and $Z := \sum_{\boldsymbol{\theta} \in \mathbb{F}_Q^N} \prod_j f_{C_j}(\boldsymbol{\theta}_{C_j})$ ensures $p(\boldsymbol{\theta})$ sums up to one. The factors f_{C_j} have local domains $\boldsymbol{\theta}_{C_j} := \{\theta_k | k \in C_j\}$, and are referred to as factor potentials [15, Section 8.3].

(A2) The network coding protocol is designed so that each y_m is a linear combination of only a subset $S_m \in \mathcal{N}$ of the observations $\mathbf{x}$, i.e.,

$$y_m = \sum_{i \in S_m} A_{m,i} x_i \tag{5.7}$$

where linear coefficients $A_{m,i} \in \mathbb{F}_Q$ are drawn randomly from a uniform distribution. The other entries $A_{m,j} = 0$ for all $j \notin S_m$, which renders the matrix $\mathbf{A}$ sparse if $|S_m| \ll N$ for all m.

Assumption (A1) subsumes the case when each cluster C_j is simply a pair of neighboring nodes. Defining $\mathcal{E}$ as the set of all pairs (n, n') of nodes where n and n' are neighbors in $\mathcal{N}$, the pmf $p(\boldsymbol{\theta})$ for the pairwise case factorizes as

$$p(\boldsymbol{\theta}) = \frac{1}{Z} \prod_{(n,n') \in \mathcal{E}} f_{nn'}(\theta_n, \theta_{n'}) \tag{5.8}$$

where Z is again the normalization constant. The choice of the subsets S_m in (A2) dictates the communication protocol used and the cost incurred. In order to save cost, individual sensors do not route their observations to the sink directly. Instead, data from all nodes in S_m are linearly combined into y_m, and then routed to the sink. This can be done efficiently by using a collection tree spanning all nodes in S_m, and rooted at a node $i \in S_m$ that is closest to the sink. Then, as explained in Appendix A, the collection procedure incurs only $|S_m| - 1$ transmissions. Let the hop-distance of node $k \in \mathcal{N}$ from the sink be denoted by h_k. Since the node in S_m that is nearest to the sink is responsible for collecting y_m from other nodes in S_m, the total communication cost of this scheme is given by $\sum_m |S_m| - 1 + \min_{k \in S_m} h_k$. In comparison, routing each observation without coding incurs a cost of $\sum_{k \in \mathcal{N}} h_k$.

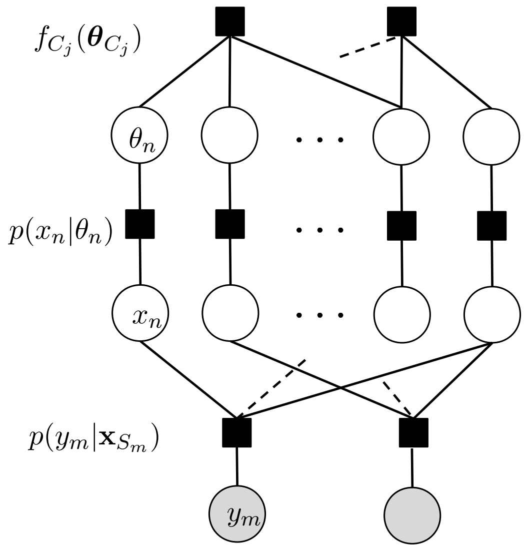

$f_{C_j}(\boldsymbol{\theta}_{C_j})$
θ_n
$p(x_n|\theta_n)$
x_n
$p(y_m|\mathbf{x}_{S_m})$
y_m

Figure 5.1: Factor graph representation of the posterior density in (5.9).

Using (A1)-(A2), it is possible to rewrite (5.5) as

$$p_\ell(\theta_\ell|\mathbf{y}) \propto \sum_{\boldsymbol{\theta}\in\mathbb{F}_Q^N\setminus\theta_\ell} \sum_{\mathbf{x}\in\mathbb{F}_Q^N} \prod_{m=1}^{M} p(y_m|\mathbf{x}_{S_m}) \prod_{n=1}^{N} p(x_n|\theta_n) \prod_{j=1}^{J} f_j(\boldsymbol{\theta}_{C_j}) \tag{5.9}$$

where $p(y_m|\mathbf{x}_{S_m}) = 1$ if $y_m = \sum_{i\in S_m} A_{m,i}x_i$, and 0 otherwise. The overall factor graph is depicted in Figure 5.1. The hollow circular nodes are the variable nodes, and denote the observed and hidden variables. The square, factor nodes correspond to the functions that appear within the summation in (5.9), and represent the relationship between the connecting variable nodes. The variable nodes representing y_m are shaded, because they are already known and need not be inferred.

The factor graph in Figure 5.1 contains cycles or loops, which generally prevents one from performing exact inference. Observe from Figure 5.1, that any cycles that may occur in the factor graph may span: (a) only the sets C_j; or (b) only the sets S_m; or (c) both C_j and S_m. Of these, cycles due to (a) are unavoidable if the prior $p(\boldsymbol{\theta})$ already has cycles and is precisely known. In principle, one could discard dependencies among some of the neighboring nodes, albeit at the expense of some model mismatch. The resulting modeling error may be justified if the performance of the sum-product algorithm improves such that overall estimation error decreases; see e.g., [162] and references therein. In practice however, only the topology of the network is specified, and a model for $p(\boldsymbol{\theta})$ must be postulated by choosing the clusters $\{C_j\}$ appropriately.

5.2.1 Cyclic Factor Graphs

If the specified topology does not admit an acyclic factor graph representation, the sum-product algorithm may still be used to find the marginal (5.9) *approximately*. The loopy version of the sum-product algorithm consists of two steps: (a) passing messages from all variable to all factor nodes, and (b) passing messages from all factor nodes to variable nodes. Denoting the variable nodes θ_n and x_n by indices ν and n, and likewise the factor nodes C_j and S_m by j and m, the expressions for messages take the following form.

$$\mu_{\nu\to j}(\theta_n) = \mu_{n\to\nu}(\theta_n)\prod_{j'\neq j}\mu_{j'\to\nu}(\theta_n) \tag{5.10a}$$

$$\mu_{j\to\nu}(\theta_n) = \sum_{\sim\{\theta_n\}} f_{C_j}(\boldsymbol{\theta}_{C_j})\prod_{n'\neq n}\mu_{\nu'\to j}(\theta_{n'}) \tag{5.10b}$$

$$\mu_{\nu\to n}(x_n) = \sum_{\theta_n\in\mathbb{F}_Q} p(x_n|\theta_n)\prod_{j'}\mu_{j'\to\nu}(\theta_n) \tag{5.10c}$$

$$\mu_{n\to\nu}(\theta_n) = \sum_{x_n\in\mathbb{F}_Q} p(x_n|\theta_n)\prod_{m'}\mu_{m'\to n}(x_n) \tag{5.10d}$$

$$\mu_{n\to m}(x_n) = \mu_{\nu\to n}(x_n)\prod_{m'\neq m}\mu_{m'\to n}(x_n) \tag{5.10e}$$

$$\mu_{m\to n}(x_n) = \sum_{\sim\{x_n\}} p(y_m|\mathbf{x}_{S_m})\prod_{n'\neq n}\mu_{n'\to m}(x_{n'}). \tag{5.10f}$$

Here the summations in (5.10b) and (5.10f) are over the vector domains $\boldsymbol{\theta}_{C_j}\in\mathbb{F}_Q^{|C_j|}\setminus\theta_n$ and $\mathbf{x}_{S_m}\in\mathbb{F}_Q^{|S_m|}\setminus x_n$ respectively. The messages $\mu_{\nu\to j}(\theta_n)$ and $\mu_{j\to\nu}(\theta_n)$ are those exchanged between C_j and θ_n, and messages $\mu_{n\to m}(x_n)$ and $\mu_{m\to n}(x_n)$ are those exchanged between S_m and x_n. For simplicity, the messages between x_n, θ_n and $p(x_n|\theta_n)$ are compacted into messages $\mu_{n\to\nu}(\theta_n)$ and $\mu_{\nu\to n}(x_n)$ (the messages to and from factors $p(x_n|\theta_n)$ are bypassed).

The algorithm starts by setting $\mu_{n\to m}(x_n) = \mu_{\nu\to j}(\theta_n) = 1$ (for all $1\leq m\leq M$, $1\leq\nu\leq N$, $1\leq j\leq J$, and $x_n,\theta_n\in\mathbb{F}_Q$), and runs for several iterations. At each iteration, the first step consists of evaluating (5.10b) followed by (5.10c), and (5.10f) followed by (5.10d), while the second step consists of evaluating (5.10a) and (5.10e). The algorithm is terminated either upon convergence, or after a fixed number of iterations and yields the approximate marginal distribution $p(\theta_n|\mathbf{y}) \propto \mu_{n\to\nu}(\theta_n)\prod_{j'}\mu_{j'\to\nu}(\theta_n)$. The complexity of this algorithm is exponential in the number of nodes in C_j and S_m (denoted respectively by $|C_j|$ and $|S_m|$), because (5.10b) and (5.10f) have $Q^{|C_j|-1}$ and $Q^{|S_m|-1}$ summands respectively. However, the number of summations and multiplications in (5.10a)–(5.10f) required at each iteration are only linear in N.

With loopy factor graphs, the sum-product algorithm does not—in general—provide any guarantees on the quality of the approximation. Related results from the coding literature suggest that short cycles typically result in poor approximations [100]. Cycles of length four may occur for instance if two sets S_1 and S_2 (or clusters C_1 and C_2) overlap in two or more nodes. Four-cycles between the clusters C_j can be avoided by using a pairwise factorization for $p(\boldsymbol{\theta})$ as in (5.8). An approximate algorithm to choose the sets $\{S_m\}$ so as to minimize the communication cost and allow no cycles among themselves is provided in Appendix 5.A. However, cycles of length eight may still occur as cluster C_j and a set S_m may share two or more nodes. The next subsection describes a scheme that allows cycles to be completely eliminated from the factor graph.

5.2.2 Acyclic Factor Graphs

As discussed earlier, for acyclic factor graphs, the sum-product algorithm is guaranteed to converge in a finite number of iterations, and finds the exact per-entry marginals $p(x_n|\mathbf{y})$ [15]. Further, some network topologies may be well-suited to an acyclic factorization of $p(\boldsymbol{\theta})$. For instance, the graph of a chain of sensors $\{1, 2, 3, \ldots, N\}$ admits a cycle-free factor graph representation with clusters of the form $C_1 = \{1, 2, 3\}$, $C_2 = \{3, 4, 5\}$, and so on.

Once the clusters C_j can be chosen to avoid cycles within themselves, other cycles can be eliminated as follows. First, let $M = J$, and set $S_j = C_j$ for all $1 \le j \le M$. Next, observe that the factor graph can be "folded" along the factor nodes $p(x_n|\theta_n)$ [cf. Fig. 5.1]. More precisely, it is always possible to combine the two variables x_n and θ_n into a single variable

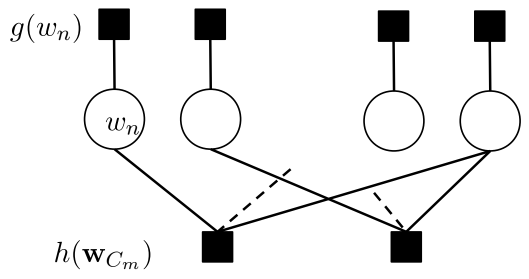


Figure 5.2: Acyclic factor graph for Section 5.2.2.

$w_n \in \mathbb{F}_Q^2$ and express the marginal pmf as

$$p(\theta_n|\mathbf{y}) \propto \sum_{\substack{\mathbf{x}\in\mathbb{F}_Q^N \\ \boldsymbol{\theta}\in\mathbb{F}_Q^N\setminus\theta_n}} \prod_{n'=1}^{N} g(w_{n'}) \prod_{m=1}^{M} h(\mathbf{w}_{C_m}) \tag{5.11}$$

where $\mathbf{w}_{C_m} := \{w_i | i \in C_m\}$, $g(w_{n'}) := p(x_{n'}|\theta_{n'})$, and $h(\mathbf{w}_{C_m}) := p(y_m|\mathbf{x}_{C_m}) f_{C_m}(\boldsymbol{\theta}_{C_m})$. The resulting factor graph is now acyclic, and an example is shown in Fig. 5.2.

The sum-product algorithm is also simpler to describe in this case, and involves passing messages $\mu_{n\to m}(w_n)$ from variable node n to factor node m, and $\mu_{m\to n}(w_n)$ from the factor node m to variable node n. These messages take the form

$$\mu_{n\to m}(w_n) = g(w_n) \prod_{m'\neq m} \mu_{m'\to n}(w_n) \tag{5.12a}$$

$$\mu_{m\to n}(w_n) = \sum_{\{w_j\in\mathbb{F}_Q^2 | j\in C_m\setminus n\}} h(\mathbf{w}_{C_m}) \prod_{n'\neq n} \mu_{n'\to m}(w_{n'}). \tag{5.12b}$$

The algorithm starts by setting $\mu_{n\to m}(w_n) = 1$ (for all $1 \leq m \leq M$, $1 \leq n \leq N$, and $w_n \in \mathbb{F}_Q^2$), runs until convergence, and yields the approximate marginal distribution $p(w_n|\mathbf{y}) \propto g(w_n)\prod_m \mu_{m\to n}(w_n)$. The variables x_n and θ_n may then be recovered by maximizing the individual marginals as described earlier. Before concluding the section, a remark is due

Remark 5.2. It is also possible for each S_m to send $L(m) > 1$ linear combinations to the sink. The availability of more than $\sum_m L(m) > M$ linear combinations at the sink can provide a better MAP estimate, though at the cost of higher communication requirement of $\sum_m |S_m| - 1 + L(m)\min_{k\in S_m} h_k$. Further in the factor graph, only the expression for $p(y_m|\mathbf{x}_{S_m})$ changes, while the structure of (5.9), and consequently the complexity of the sum-product algorithm remains the same. Varying the values of $L(m)$ across clusters thus provides a low-complexity method of exploring the cost-performance tradeoff.

5.3 Error Exponents

In this section, bounds on the probability of error are evaluated for the block MAP estimator (5.2). For simplicity, bounds are first derived in Section 5.3.1 for the factor graph representation of 5.2.2 in which the resulting factor graph is acyclic and thus easier to handle. The bounds in Section 5.3.2 on the other hand, require a pairwise correlation model, but are valid even if the resulting factor graph is cyclic. Both subsections assume that the observation noise is zero, i.e., $p(x|\theta) = \mathbb{1}_{x=\theta}$.

5.3.1 Acyclic Factor Graphs with General Correlation Model

As discussed in Section 5.2.2, nodes are divided into overlapping clusters $\{C_m\}_{m=1}^{M}$. Correlated observations of each cluster are sent to the sink after being linearly combined into a single symbol in $\mathbb{F}_Q$. The clusters are constructed in such a way that the resulting factor graph is acyclic.

The sink node can tolerate a limited amount of distortion in the reconstructed $\hat{\mathbf{x}}$ ($:= \hat{\boldsymbol{\theta}}$). Define the cluster-Hamming distortion metric $D_H(\mathbf{x}, \mathbf{x}')$ between two vectors $\mathbf{x}$ and $\mathbf{x}'$ as the fraction of clusters over which the two vectors differ, i.e.,

$$D_H(\mathbf{x}, \mathbf{x}') = \frac{|\{m | \mathbf{x}_{C_m} \neq \mathbf{x}'_{C_m}\}|}{M}. \tag{5.13}$$

The probability of error P_e is the average probability that the distortion between the observed vector $\mathbf{x}$ and the decoded vector $\hat{\mathbf{x}}$ is greater than a tolerable level d, i.e.,

$$P_e = \sum_{\mathbf{x} \in \mathbb{F}_Q^N} \text{Pr}(D_H(\hat{\mathbf{x}}, \mathbf{x}) \geq d | \mathbf{x}) p(\mathbf{x}). \tag{5.14}$$

The conditional error probability $\text{Pr}(D_H(\hat{\mathbf{x}}, \mathbf{x}) \geq d | \mathbf{x})$ can be bounded as shown in the following lemma; see Appendix 5.B for the proof.

Lemma 5.1. *The conditional probability that the distortion $D_H(\hat{\mathbf{x}}, \mathbf{x})$ exceeds a tolerable threshold d, can be bounded as*

$$Pr(D_H(\hat{\mathbf{x}}, \mathbf{x}) \geq d | \mathbf{x}) \leq \sum_{\substack{\mathbf{z} \in \mathbb{F}_Q^N, D_H(\mathbf{z}, \mathbf{x}) \geq d \\ p(\mathbf{z}) \geq p(\mathbf{x})}} Q^{-dM}. \tag{5.15}$$

The intuition behind Lemma 5.1 comes from the observation that if two vectors $\mathbf{x}$ and $\mathbf{z}$ differ over a single cluster, the probability that a random $\mathbf{A}$ satisfies $\mathbf{Ax} = \mathbf{Az}$ is exactly $1/Q$. Thus, when the two vectors differ over dM clusters (corresponding to a distortion d), $\mathbf{A}$ satisfies $\mathbf{Ax} = \mathbf{Az}$ with probability Q^{-dM}.

Interestingly, it is possible to obtain a compact form of the bound in (5.15) when $p(\mathbf{x})$ has an acyclic factor graph representation. Define the cluster graph $\mathcal{G}_C = (\mathcal{V}_C, \mathcal{E}_C)$ as the undirected graph formed by the set of M nodes $\mathcal{V}_C$, representing the clusters $\{C_m\}_{m=1}^M$, and the set of edges $\mathcal{E}_C$, connecting pairs of overlapping clusters. For the pmf $p(\mathbf{x})$ to be an acyclic factor graph, it is necessary that the cluster graph is also acyclic, or equivalently tree-shaped. In this case, it is always possible to factor $p(\mathbf{x})$ in terms of the individual and pairwise cluster pmfs as follows [163]

$$p(\mathbf{x}) = \prod_{m \in \mathcal{V}_C} p_m(\mathbf{x}_m) \prod_{(m,m') \in \mathcal{E}_C} \frac{p_{m,m'}(\mathbf{x}_m, \mathbf{x}_{m'})}{p_m(\mathbf{x}_m) p_{m'}(\mathbf{x}_{m'})} \tag{5.16}$$

where node m represents cluster C_m, $\mathbf{x}_m$ represents set $\mathbf{x}_{C_m}$, and $p_m(\mathbf{x}_m)$ and $p_{m,m'}(\mathbf{x}_m, \mathbf{x}_{m'})$ represent, respectively, the joint pmfs over $\mathbf{x}_{C_m}$ and $\mathbf{x}_{C_m \cup C_{m'}}$. Supposing identical clusters, the subscripts m and m' can be removed from the pmfs. Since $\mathcal{G}_C$ is a tree, it is possible to choose any cluster C_1 as its root, reorder pairs (m, m') appropriately, and express (5.16) as

$$p(\mathbf{x}) = p(\mathbf{x}_1) \prod_{(m,m') \in \mathcal{E}_C} p(\mathbf{x}_m | \mathbf{x}_{m'}). \tag{5.17}$$

Because of the Markov property, the variables $\mathbf{x}_m$ depend on the variables $\mathbf{x}_{m'}$ only through the variables common to both C_m and $C_{m'}$, i.e., $p(\mathbf{x}_m|\mathbf{x}_{m'}) = p(\mathbf{x}_m|\mathbf{x}_{C_m \cap C_{m'}})$. Further, as discussed in Section 5.2.1, for the factor graph to be acyclic, two clusters can overlap in at most one variable, meaning that $|C_m \cap C_{m'}| = 1$. With $x_1 \in C_1$, and $p(\mathbf{x}_1) = p(x_1)p(\mathbf{x}_{C_1 \setminus \{1\}}|x_1)$, (5.17) can be rearranged as

$$p(\mathbf{x}) = p(x_1) \prod_{m \in \mathcal{V}_C} p(\mathbf{x}_{C_m \setminus j_m} | x_{j_m}) \tag{5.18}$$

where j_m is some node in the set C_m.

With these assumptions, it is possible to manipulate $p(\mathbf{x})$ using the method of types [37], as detailed next. If each cluster has exactly K nodes, the pmf $p(\mathbf{x}_{C_m \backslash j_m} | x_{j_m})$ takes at most Q^K values, also referred to as types. Defining $p_{q,\boldsymbol{i}} := p(\mathbf{x}_{C_m \backslash j_m} = \boldsymbol{i} | x_{j_m} = q)$, and $\ell_{q,\boldsymbol{i}} := |\{m \in \mathcal{V}_C | \mathbf{x}_{C_m \backslash j_m} = \boldsymbol{i}, x_{j_m} = q\}|$, the prior pmf in (5.18) can be expressed as

$$p(\mathbf{x}) = \frac{1}{Q} \prod_{q=1}^{Q} \prod_{\boldsymbol{i} \in \mathcal{I}_q} p_{q,\boldsymbol{i}}^{\ell_{q,\boldsymbol{i}}(\mathbf{x})} \tag{5.19}$$

where the set $\mathcal{I}_q := \{\boldsymbol{i} | p_{q,\boldsymbol{i}} \neq 0\}$, and $|\mathcal{I}_q| \leq Q$. Pmf $p(\mathbf{x})$ now depends on $\mathbf{x}$ through its type $\{\ell_{q,\boldsymbol{i}}(\mathbf{x})\}_{q,\boldsymbol{i}}$, compactly denoted by the $Q^{K-1} \times Q$ type matrix $\mathbf{L}(\mathbf{x})$, with entries

$$[\mathbf{L}(\mathbf{x})]_{q,\boldsymbol{i}} := \begin{cases} \ell_{q,\boldsymbol{i}}(\mathbf{x}) & p_{q,\boldsymbol{i}} \neq 0 \\ 0 & \text{otherwise} \end{cases} . \tag{5.20}$$

Conversely, given a type matrix $\boldsymbol{\Lambda}$, define $\mathcal{T}(\boldsymbol{\Lambda}) := \{\mathbf{x} | \mathbf{L}(\mathbf{x}) = \boldsymbol{\Lambda}\}$ as the set of vectors $\mathbf{x}$ that have the same type. In order to state the next result, a few definitions are needed. Let p_q be the pmf induced by $p_{q,\boldsymbol{i}}$ for fixed q, and $\varphi_q(\mathbf{x})$ be the pmf induced by $\ell_{q,\boldsymbol{i}}(\mathbf{x})/\ell_q(\mathbf{x})$, where $\ell_q(\mathbf{x}) := \sum_{\boldsymbol{i}} \ell_{q,\boldsymbol{i}}(\mathbf{x})$. Also define $H(\varphi_q(\mathbf{x}))$ as the entropy of the pmf $\varphi_q(\mathbf{x})$, and $D(\varphi_q(\mathbf{x}) \| p_q)$ as the Kullback-Leibler (KL) divergence between pmfs $\varphi_q(\mathbf{x})$ and p_q [35]. Then the following lemma holds; see Appendix 5.B for the proof.

Lemma 5.2. *The pmf $p(\mathbf{x})$ in (5.18) can be written as*

$$p(\mathbf{x}) = \frac{1}{Q} 2^{-M(H_{\mathbf{L}(\mathbf{x})} + D_{\mathbf{L}(\mathbf{x}),p})} \tag{5.21}$$

where $H_{\mathbf{L}(\mathbf{x})} := \sum_{q=1}^{Q} \frac{\ell_q(\mathbf{x})}{M} H(\varphi_q(\mathbf{x}))$ and $D_{\mathbf{L}(\mathbf{x}),p} := \sum_{q=1}^{Q} \frac{\ell_q(\mathbf{x})}{M} D(\varphi_q(\mathbf{x}) \| p_q)$ are the average entropy and divergence operators.

Let $\mathcal{L}$ denote the set of all possible types. Then, summing over all possible values of $\mathbf{x}$ is equivalent to summing over all types $\boldsymbol{\Lambda} \in \mathcal{L}$, and summing over all vectors $\mathbf{x} \in \boldsymbol{\Lambda}$ [37]. Using

Lemma 5.2 in (5.14), it can be seen that

$$\begin{aligned} P_e &= \frac{1}{Q}\sum_{\mathbf{\Lambda}\in\mathcal{L}}\sum_{\mathbf{x}\in\mathcal{T}(\mathbf{\Lambda})} P_{e|\mathbf{\Lambda}}2^{-M(H_{\mathbf{\Lambda}}+D_{\mathbf{\Lambda},p})} && (5.22a)\\ &= \frac{1}{Q}\sum_{\mathbf{\Lambda}\in\mathcal{L}} P_{e|\mathbf{\Lambda}}|\mathcal{T}(\mathbf{\Lambda})|2^{-M(H_{\mathbf{\Lambda}}+D_{\mathbf{\Lambda},p})} && (5.22b)\\ &\leq \frac{1}{Q}\sum_{\mathbf{\Lambda}\in\mathcal{L}} P_{e|\mathbf{\Lambda}}2^{-MD_{\mathbf{\Lambda},p}} && (5.22c)\\ &\leq \frac{1}{Q}(M+1)^{QK}P_{e|\mathbf{\Lambda}}2^{-MD_{\mathbf{\Lambda},p}} && (5.22d)\end{aligned}$$

where $P_{e|\mathbf{\Lambda}} :=$ Pr$(D_H(\hat{\mathbf{x}},\mathbf{x}) \geq d|\mathbf{x}\in\mathcal{T}(\mathbf{\Lambda}))$; the inequality in (5.22c) makes use of the bound $|\mathcal{T}(\mathbf{\Lambda})| \leq Q2^{MH_{\mathbf{\Lambda}}}$ (see Appendix 5.B); and (5.22d) considers the fact that the total number of types can be bounded as $|\mathcal{L}| \leq (M+1)^{QK}$.

The bound in (5.22d) thus depends on whether or not $D_{\mathbf{\Lambda},p}$ is zero. Indeed, if $[\mathbf{\Lambda}]_{q,\boldsymbol{i}} = p_{q,\boldsymbol{i}}$ for all q and $\boldsymbol{i}$, it holds that $D_{\mathbf{\Lambda},p} = 0$. Such a type will be henceforth denoted as $\mathbf{\Lambda}^*$, and any $\mathbf{x}\in\mathcal{T}(\mathbf{\Lambda}^*)$ will be referred to as *typical*. It can be seen that except for typical vectors $\mathbf{x}$, P_e goes to zero as M increases because the first term in (5.22d) grows only polynomially in M. The following proposition summarizes this result.

Proposition 5.1. *For large M, the error probability P_e in* (5.22d) *goes to zero for non-typical* $\mathbf{x}$, *i.e.,* $\mathbf{x}\notin\mathcal{T}(\mathbf{\Lambda}^*)$*, and goes to* $\frac{1}{Q}(M+1)^{QK}P_{e|\mathbf{\Lambda}^*}$ *for typical* $\mathbf{x}$, *i.e.,* $\mathbf{x}\in\mathcal{T}(\mathbf{\Lambda}^*)$.

In other words, the exponent of the conditional probability $P_{e|\mathbf{\Lambda}^*}$ for the typical $\mathbf{x}$ dominates the overall error probability. In order to derive bounds on $P_{e|\mathbf{\Lambda}^*}$, note again that the summation in Lemma 5.1 (over $\mathbf{z}$) can be expressed as summation over types $\mathbf{\Omega}\in\mathcal{L}$ and vectors in each type $\mathbf{z}\in\mathcal{T}(\mathbf{\Omega})$, yielding

$$\text{Pr}(D_H(\hat{\mathbf{x}},\mathbf{x}) \geq d|\mathbf{x}\in\mathcal{T}(\mathbf{\Lambda}^*)) \leq \sum_{\substack{\mathbf{\Omega}\in\mathcal{L}\\ p(\mathbf{z})\geq p(\mathbf{x})}}\sum_{\mathbf{z}\in\mathcal{T}(\mathbf{\Omega})} Q^{-dM}. \tag{5.23}$$

Given that $\mathbf{x}\in\mathcal{T}(\mathbf{\Lambda}^*)$, the condition $p(\mathbf{z}) \geq p(\mathbf{x})$ can simply be expressed as $H_{\mathbf{\Omega}} + D_{\mathbf{\Omega},p} \leq H_{\mathbf{\Lambda}^*} + D_{\mathbf{\Lambda}^*,p} = H_{\mathbf{\Lambda}^*}$. Replacing the summation over $\mathbf{z}\in\mathcal{T}(\mathbf{\Omega})$ by the bound $|\mathcal{T}(\mathbf{\Omega})| \leq$

$Q2^{MH_{\mathbf{\Omega}}}$ [cf. (5.22b) and (5.22c)] it follows that

$$\begin{aligned}\Pr(D_H(\hat{\mathbf{x}}, \mathbf{x}) \geq d | \mathbf{x} \in \mathcal{T}(\mathbf{\Lambda}^*)) &\leq Q^2 \sum_{\substack{\mathbf{\Omega} \in \mathcal{L} \\ H_{\mathbf{\Omega}} + D_{\mathbf{\Omega},p} \leq H_{\mathbf{\Lambda}^*}}} 2^{-M(d \log Q - H_{\mathbf{\Omega}})} \\ &\leq Q \sum_{\mathbf{\Omega} \in \mathcal{L}} 2^{-M(d \log Q - H_{\mathbf{\Lambda}^*})} \\ &\leq Q |\mathcal{L}| 2^{-M(d \log Q - H_{\mathbf{\Lambda}^*})}. \end{aligned} \tag{5.24}$$

As observed earlier, since $|\mathcal{L}|$ is only polynomial in M, so that the overall P_e is dominated only by the exponential term. Finally, the conditional probability is always less than or equal to one, so the exponent should always be negative. The following proposition summarizes the result.

Proposition 5.2. *For sufficiently large M, the error exponent of the probability of error P_e is bounded as $E \geq [d \log Q - H_{\mathbf{\Lambda}^*}]^+$, where $[\mathbf{\Lambda}^*]_{q,\boldsymbol{i}} = p_{q,\boldsymbol{i}}$ for all $\boldsymbol{i} \in \mathcal{I}_q$, $1 \leq q \leq Q$.*

It can thus be observed that larger values of Q yield smaller probabilities of error. Intuitively, $\log Q$ is the number of observed bits at each sensor, $d \log Q$ is the number of bits per-sensor that need to be reconstructed correctly at the sink, and $H_{\mathbf{\Lambda}^*}$ represents the total number of (uncorrelated) information bits observed by the sensor network as a whole. If the entropy $H_{\mathbf{\Lambda}^*}$ is small, it means that sensor observations are highly correlated. This happens for instance, when the transition probability $p_{q,\boldsymbol{i}}$ is close to 1 if all entries of $\boldsymbol{i}$ are equal to q. In this case, a smaller Q can also be used to recover information with smaller allowable distortion.

5.3.2 Cyclic Factor Graphs with Pairwise Correlation Model

This subsection derives bounds on the probability of error for cyclic graphs assuming the pairwise correlation model in (5.8). Define the graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with the set of nodes $\mathcal{N}$ representing the sensors, and the edges $\mathcal{E}$ connecting neighboring nodes. In this case, the pmf of $\mathbf{x}$ can be expressed as a product of factors along a spanning tree $\mathcal{E}_T$, and the rest of the edges $\bar{\mathcal{E}}_T$ [cf. (5.16)]

$$p(\mathbf{x}) = \frac{1}{W} \prod_v p_v(x_v) \prod_{(v,w) \in \mathcal{E}_T} \frac{p_{v,w}(x_v, x_w)}{p_v(x_v) p_w(x_w)} \prod_{(v,w) \in \bar{\mathcal{E}}_T} \frac{p_{v,w}(x_v, x_w)}{p_v(x_v) p_w(x_w)} \tag{5.25}$$

where $W := \sum_{\mathbf{x}} \prod_v p_v(x_v) \prod_{(v,w)\in\mathcal{E}} \frac{p_{v,w}(x_v,x_w)}{p_v(x_v)p_w(x_w)}$ and $\mathcal{E} = \mathcal{E}_T \cup \bar{\mathcal{E}}_T$, with $\mathcal{E}_T \cap \bar{\mathcal{E}}_T = \emptyset$ is the set of all edges representing the graphical model. Notice that if $\bar{\mathcal{E}}_T = \emptyset$ then $W = 1$ and the model in (5.25) boils down to the one in (5.16). Assuming identical joint probabilities so that $p(x_v, x_w) := p_{v,w}(x_v, x_w)$, and uniform prior probabilities $p_v(x_v) = 1/Q$, $p(\mathbf{x})$ can be compactly written in terms of conditional edge transition probabilities

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{(v,w)\in\mathcal{E}_T} p(x_v|x_w) \prod_{(v,w)\in\bar{\mathcal{E}}_T} p(x_v|x_w) \tag{5.26}$$

where now $Z := \prod_{(v,w)\in\mathcal{E}} p(x_v|x_w) \prod_{(v,w)\in\bar{\mathcal{E}}_T} p(x_v|x_w)$ replaces W in (5.25). Note that in this model $p(x_1)$ is not explicitly shown since the normalization constant Z is needed anyway. The types can now be defined as the Q^2 values the conditional pmf $p(x_v|x_w)$ takes. Defining $p_{q,i}$ and $\ell_{q,i}(\mathbf{x})$ in a similar manner as in Section 5.3.1, it holds that

$$\ell_{q,i}(\mathbf{x}) = \ell^T_{q,i}(\mathbf{x}) + \bar{\ell}^T_{q,i}(\mathbf{x}) \tag{5.27}$$

where $\ell^T_{q,i}(\mathbf{x})$ counts the number of transitions of (q,i)-th type for edges in $\mathcal{E}_T$, whereas $\bar{\ell}^T_{q,i}(\mathbf{x})$ counts the transitions for edges in $\bar{\mathcal{E}}_T$. Proceeding as in Lemma 5.2, $p(\mathbf{x})$ in (5.26) can be written as

$$p(\mathbf{x}) = \frac{1}{Z} 2^{-|\mathcal{E}_T|(H_{\mathbf{L}_T(\mathbf{x})} + D_{\mathbf{L}_T(\mathbf{x}),p})} 2^{-|\bar{\mathcal{E}}_T|(H_{\bar{\mathbf{L}}_T(\mathbf{x})} + D_{\bar{\mathbf{L}}_T(\mathbf{x}),p})} \tag{5.28}$$

where $H_{\mathbf{L}_T(\mathbf{x})}$, $D_{\mathbf{L}_T(\mathbf{x}),p}$, $H_{\bar{\mathbf{L}}_T(\mathbf{x})}$ and $D_{\bar{\mathbf{L}}_T(\mathbf{x}),p}$ are defined as in Lemma 5.2, normalizing counts over $|\mathcal{E}_T|$ and $|\bar{\mathcal{E}}_T|$. Using this representation for $p(\mathbf{x})$, the next proposition connects the conditional error probability of cyclic graphs with respect to acyclic ones; see Appendix 5.B for the proof.

Proposition 5.3. *For large $|\mathcal{E}_T|$, the error probability P_e in* (5.22d) *with the prior $p(\mathbf{x})$ as in* (5.28) *goes to zero for non-typical $\mathbf{x}$, i.e., $\mathbf{x} \notin \mathcal{T}(\mathbf{\Lambda}^*_T)$, and goes to $P_{e|\mathbf{\Lambda}^*_T}$ for the typical $\mathbf{x} \in \mathcal{T}(\mathbf{\Lambda}^*_T)$.*

The consequence of this proposition is that the error probability of cyclic $p(\mathbf{x})$ is governed by the error probability of any underlying tree. In fact, it will next be shown that for any $\mathbf{x}$ (typical or not) the conditional error probability with acyclic $p(\mathbf{x})$ can be bounded by the same bound benchmarking the performance of any underlying tree in the graph. This is possible by appropriately bounding the conditional error probability $\Pr(D_H(\hat{\mathbf{x}}, \mathbf{x}) \geq d|\mathbf{x})$ as shown next

$$\begin{aligned}\Pr(D_H(\hat{\mathbf{x}}, \mathbf{x}) \geq d|\mathbf{x}) &\leq \sum_{\substack{\mathbf{\Omega}_T \in \mathcal{L} \\ H_{\mathbf{\Omega}_T} + D_{\mathbf{\Omega}_T,p} \leq H_{\mathbf{\Lambda}_T^*}}} \sum_{\substack{\mathbf{z} \in \mathcal{T}(\mathbf{\Omega}_T) \\ H_{\bar{\mathbf{\Omega}}_T} + D_{\bar{\mathbf{\Omega}}_T,p} \leq H_{\bar{\mathbf{\Lambda}}_T} + D_{\bar{\mathbf{\Lambda}}_T,p}}} Q^{-dN} \\ &\leq \sum_{\substack{\mathbf{\Omega}_T \in \mathcal{L} \\ H_{\mathbf{\Omega}_T} + D_{\mathbf{\Omega}_T,p} \leq H_{\mathbf{\Lambda}_T^*}}} |\mathcal{T}(\mathbf{\Omega}_T)| Q^{-dN} \qquad (5.29)\end{aligned}$$

where the first inequality holds because the constraint $p(\mathbf{z}) \geq p(\mathbf{x})$, which as per (5.28) is equivalent to $|\mathcal{E}_T|(H_{\mathbf{\Omega}_T} + D_{\mathbf{\Omega}_T,p}) + |\bar{\mathcal{E}}_T|(H_{\bar{\mathbf{\Omega}}_T} + D_{\bar{\mathbf{\Omega}}_T,p}) \leq |\mathcal{E}_T| H_{\mathbf{\Lambda}_T^*} + |\bar{\mathcal{E}}_T|(H_{\bar{\mathbf{\Lambda}}_T} + D_{\bar{\mathbf{\Lambda}}_T,p})$, can be split into two constraints, one for the tree-types $\mathbf{\Omega}_T$ and $\mathbf{\Lambda}_T^*$ and another for the non-tree types $\bar{\mathbf{\Omega}}_T$ and $\bar{\mathbf{\Lambda}}_T$. Using again the bound $|\mathcal{T}(\mathbf{\Omega}_T)| \leq Q 2^{|\mathcal{E}_T| H_{\mathbf{\Omega}_T}}$, (5.29) can be bounded as

$$\Pr(D_H(\hat{\mathbf{x}}, \mathbf{x}) \geq d|\mathbf{x}) \leq Q \sum_{\substack{\mathbf{\Omega}_T \in \mathcal{L} \\ H_{\mathbf{\Omega}_T} + D_{\mathbf{\Omega}_T,p} \leq H_{\mathbf{\Lambda}_T^*}}} 2^{-|\mathcal{E}_T|(\log Q - H_{\mathbf{\Omega}_T})} \qquad (5.30)$$

coincides with the bound obtained for acyclic graphs [cf. (5.24)]. One remark is now in order.

Remark 5.3. Although error probability bounds are identical for cyclic and acyclic cases, this holds true only for the exact MAP estimation. As the sum-product algorithm applied to a cyclic factor graphs yields approximate probabilities, its performance may be worse than that of acyclic graphs.

Remark 5.4. The error probability bounds derived in this section provide a useful quanitative description of the interplay between different parameters in a sensor network. However, care should be taken when applying them to a real sensor network, especially with regards to the following assumptions.

1. The bounds derived here are tight only if M is sufficiently large. Their applicability for predicting the performance of small or moderate-sized networks is therefore limited.

2. The present analysis ignores modeling error, which is otherwise a major issue in distributed compression implementations. For example, a simple correlation model, such as the one postulated in (5.6) may not be sufficient for a large network.

5.4 Simulations

5.4.1 Sum-Product on Acyclic Factor Graphs

In order to test the performance of the MAP estimator, the network-compression protocol for acyclic factor graphs, developed in Section 5.2.2, was tested on two different topologies. First in order to test the error exponents derived in Section 5.3.1, consider a simple sensor network consisting of a chain graph of the form $\{1, 2, 3, 4, \dots, N\}$. Sets C_j and S_m are both chosen to be of the form $\{1, 2, 3\}$, $\{3, 4, 5\}$, $\{5, 6, 7\}$ $\dots$, and the factor graph of Figure 5.2 is used. Given the value of Q (=4 in this case), sensors observe random integers between 1 and Q, which are then mapped to the Q elements of $\mathbb{F}_Q$. The integer label of an element $x \in \mathbb{F}_Q$ is henceforth denoted by $I(x)$, and likewise for a vector $\mathbf{x}$. Observation errors are ignored for simplicity, and the sensor observations within each cluster are assumed to follow the pmf

$$p(\mathbf{x}_{C_j}) \propto \exp(-\alpha(I(x_{j_{\max}}) - I(x_{j_{\min}}))) \tag{5.31}$$

where $j_{\max} := \arg\max_{k \in C_j} I(x_k)$ and $j_{\min} := \arg\min_{k \in C_j} I(x_k)$. Clearly this pmf encourages observations within a cluster to be close to each other. Since the factorization of $p(\mathbf{x})$ includes no cycles, vectors $\mathbf{x}$ can be sampled in a sequential manner; see e.g., [15, Chap. 8].

Figure 5.3 plots error probability P_e [cf. (5.14)] as a function of the tolerable distortion level d for $\alpha = 3$ and different values of M. According to the error exponent derived in Proposition 2, $P_e \to 0$ for $M \to \infty$, for all values of $d > H_{\mathbf{\Lambda}^*}/\log_2 Q$. Observe that the derived bound is loose, as P_e becomes very small even for values of d below $H_{\mathbf{\Lambda}^*}/\log_2 Q$ (depicted by the vertical line) and for $M \geq 50$. Nevertheless, the exponent is a good indicator of the distortion at which low P_e can be obtained at a moderate value of M.

In the context of sensor networks, it is also interesting to quantify the ℓ_0- and ℓ_1-norm of the estimation error. In particular, $e_0 := \left\| I(\boldsymbol{\theta}) - I(\hat{\boldsymbol{\theta}}) \right\|_0 / N$ represents the fraction of entries that are decoded incorrectly, and is upper bounded by $D_H(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}_0)MK/N$ [cf. Section 5.3.1]. The per-entry difference between the observed and decoded vectors, given by $e_1 := \left\| I(\boldsymbol{\theta}) - I(\hat{\boldsymbol{\theta}}) \right\|_1 / N$ is also important since the sensor observations are derived from continuous valued data, and errors with small magnitudes may be tolerable. Towards this end, consider the topology depicted in Figure 5.4 where the clusters used to partition the sensor

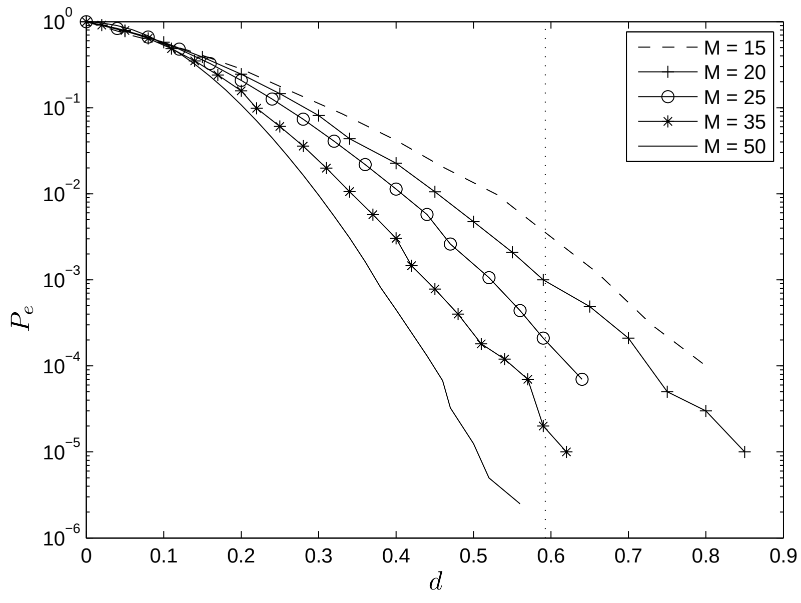


Figure 5.3: Probability of error when a distortion d can be tolerated at the sink, for different values of M, and $\alpha = 3$. The vertical line shows the distortion above which $P_e \to 0$ whenever $M \to \infty$.

nodes are also shown. Figure 5.5 shows the two measures of estimation error against α, which signifies the level of intra-cluster correlation, for $Q = 16$. As expected, both error norms decrease as α increases. Interestingly, the e_1 error is close to the e_0 error, suggesting that all decoding errors have small magnitudes. Note that with $Q = 16$, the per-entry error $e_1 \approx 0.1$ is equivalent to having each entry of $\hat{\boldsymbol{\theta}}$ incur an error of about 0.63%.

Next, the impact of varying communication cost [cf. Section 5.2.2] on the performance of the proposed algorithm is studied. This is achieved by varying the values of $L(m)$, which changes both the communication cost as well as the compression ratio. Figure 5.6 shows this compression-performance trade-off for $Q = 16$, and $\alpha = 2$. The communication cost is expressed as the percentage of the cost incurred when sending all observations through the shortest path tree. Such graphs can be used by the network designer to efficiently find the communication cost incurred for different levels of tolerable estimation errors.

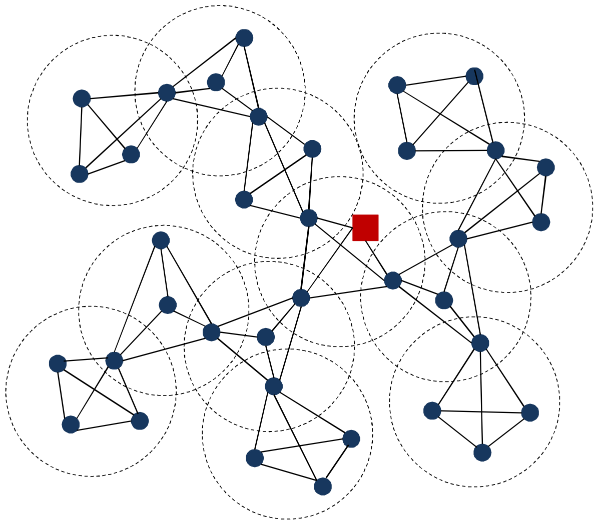


Figure 5.4: Sensors within the dotted circles are assumed correlated, with edges denoting communication links. All nodes within each cluster collect data at one of the nodes, and send it to the sink through the shortest path.

5.4.2 Performance Evaluation with the Sensorscope Dataset

The proposed network-compressive scheme is tested on the dataset available from the Sensorscope LUCE Project [147]. The LUCE deployment consists of a sensor network, shown in Figure 5.8, over a university campus measuring environmental quantities such as temperature, humidity, wind speed, etc. Only a part of the deployed network is considered here, as not all sensors were active at all times.

Temperature readings are quantized and mapped to integers between 1 and Q, and then to elements of $\mathbb{F}_Q$ to form the vector $\boldsymbol{\theta}$. The pmf $p(x_n|\theta_n)$ modeling the observation error is

$$p(x_n|\theta_n) = \begin{cases} 0.01 & I(x_n) = I(\theta_n) \pm 1, I(\theta_n) \neq 1, Q \\ 0.02 & I(x_n) = 2, I(\theta_n) = 1 \text{ or, } I(x_n) = Q-1, I(\theta_n) = Q \\ 0.98 & I(x_n) = I(\theta_n) \end{cases} \tag{5.32}$$

which roughly translates to a probability of error of 0.01, except when the sensor observes extreme values. The network is modeled using the factor graph of Figure 5.1, with hidden variables following the pairwise correlation model (5.8). The factors are chosen as $f(\theta_k, \theta_\ell) :=$

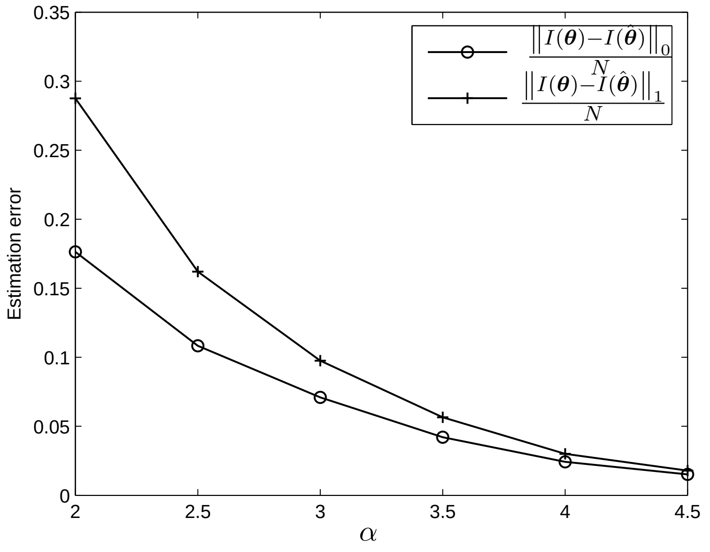


Figure 5.5: Estimation error for different levels of cluster correlation evaluated for $Q = 16$. For each cluster, $x_{\max} = \max_{k\in C_j} I(x_k)$ and $x_{\min} = \min_{k\in C_j} I(x_k)$, and the joint pmf $p(\boldsymbol{\theta}_{C_j}) \propto e^{-\alpha(x_{\max}-x_{\min})}$.

$\exp(-2|I(\theta_k) - I(\theta_\ell)|)$ for all the edges.

For this model, not all neighbors can be included in the edge set $\mathcal{E}$, or it leads to a large number of short cycles in the corresponding factor graph; see e.g., [22]. To avoid this situation, the k-nearest neighbor (kNNG) graph is used. Cycles in the graph are minimized for smaller values of k, so the smallest possible k that yields a connected kNNG is employed.

The sum-product algorithm as described in Section 5.2 is used. Towards this end, the sets S_m are chosen using the algorithm in Appendix 5.A, with at most three nodes per cluster. Node 1 is assumed to be the sink, and all sets S_m that contain node 1 send their data as is to node 1. Figure 5.8 shows the estimation error (as described in Section 5.4.1) against the communication cost. The different levels of communication costs arise from different number of linear combinations sent by the clusters. As expected, even in the cyclic case, the estimation error goes down as the communication cost is allowed to increase. However, the estimation error is higher here compared to that in the synthetic data, because: (a) the sum-product algorithm does not always converge, or converges to incorrect estimates; and (b) $p(\boldsymbol{\theta})$

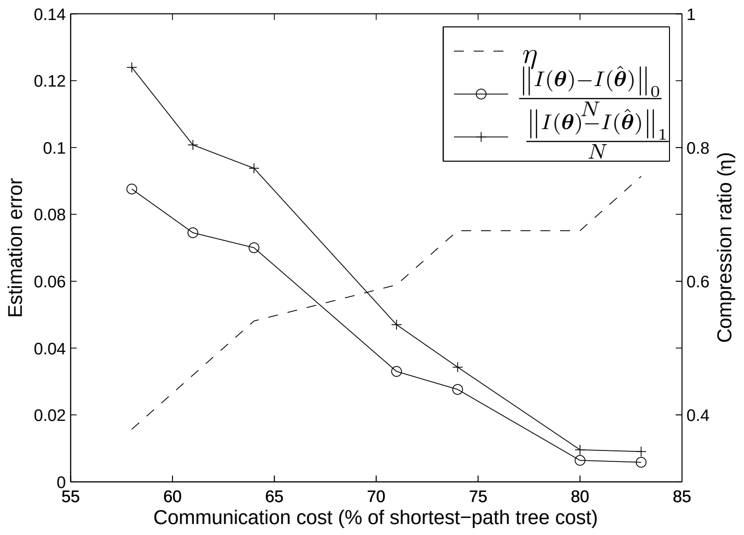


Figure 5.6: Estimation error for different levels of compression, plotted against the communication cost. As communication cost increases, more linear combinations can be sent to the sink per cluster, yielding higher compression ratios but lower estimation errors.

and $p(\mathbf{x}|\boldsymbol{\theta})$ are no longer the true probabilities representative of the real data. Nevertheless, with 75% communication cost, while about 15% entries are incorrectly estimated, the per-entry error is only about 1.25%[1].

5.5 Conclusions

A network-compressive coding scheme for sensor networks was developed. Probabilistic relationships among sensor observations were exploited to formulate the MAP estimation problem within the Bayesian inference framework. The sum-product algorithm was then utilized to perform (approximate) low-complexity decoding, with reduced communication overhead. Error exponents and simulation results were provided to delineate, quantify, and test the interplay between the estimation error, tolerable distortion, alphabet size, and communication cost.

[1]At $Q = 16$, the error $e_1 \approx 0.2$ is equivalent to a per-entry error of about 1.25%.

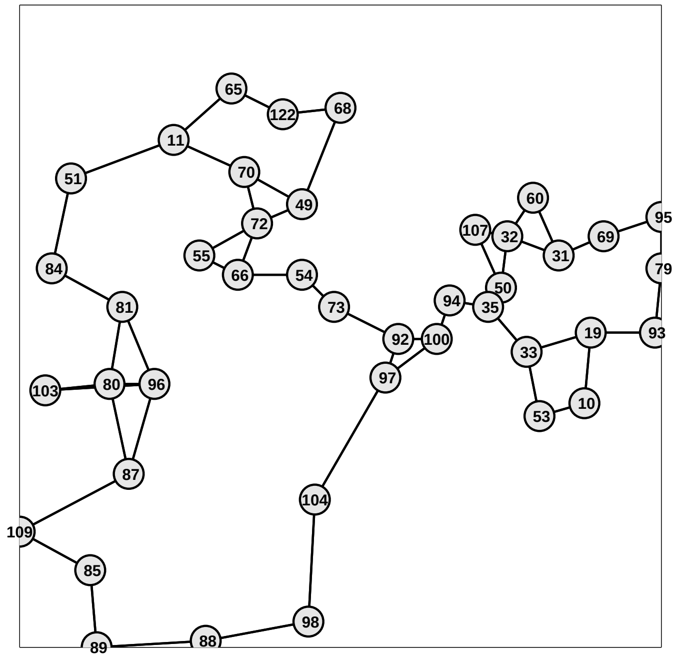


Figure 5.7: Sensor network used for the simulations. Node IDs correspond to those in the Sensorscope dataset.

5.A Choosing the Sets $\{S_m\}$ and $\{C_j\}$

Consider first choosing the sets $\{S_m\}$ such that the factor sub-graph formed by them is cycle-free, and the total communication cost [cf. Section 5.2] is minimized. Given the communication graph of the sensor network, the problem is combinatorial, and only an approximate algorithm is provided here. To ensure that the sum-product algorithm runs efficiently, it is assumed that $2 \leq |S|_m \leq K$.

Note first that the observations from all nodes in S_m are collected at a node $i \in S_m$ and then sent to the sink. This collection procedure requires all nodes $k \in S_m \setminus \{i\}$ to at least transmit once, and thus incur a total cost of at least $|S_m| - 1$. If the subgraph formed by nodes in S_m is connected, it can be shown that the collection cost of $|S_m| - 1$ is also achievable. Consider the collection tree rooted at node i, and connected to other nodes in S_m. As shown in Figure, each node requires only one transmission. Specifically, the leaf nodes transmit their observations uncoded, while the intermediate nodes transmit the linear combination formed by

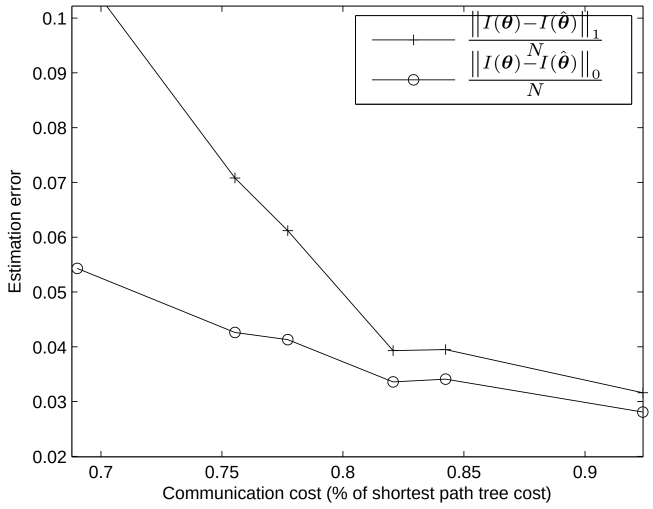


Figure 5.8: Estimation error vs. the communication cost. More communication allows more linear combinations to be sent to the sink per cluster, yielding lower estimation errors.

their own observation and the received symbol. Therefore, given set S_m that forms a connected subgraph, a graph traversal algorithm (such as breadth-first or depth-first search) can be used to find a collection tree that is rooted at the node closest to the sink.

Having recognized that the subgraph formed by nodes in each S_m must be connected, the following algorithm adds a new S_m per iteration, while maintaining the acyclic nature of the factor sub-graph F formed by $\{S_m\}$.

1. Let $\mathcal{R}$ be the set of nodes that have already been added, and initialize $\mathcal{R} \leftarrow \emptyset$. Also initialize factor graph F with variables nodes $\mathcal{N}$.

2. Unless $\mathcal{R} = \mathcal{N}$, repeat

   (a) choose S_m as any connected subgraph with at most K nodes, of which one is from $\mathcal{R}$ and others from $\mathcal{N} \setminus \mathcal{R}$; and

   (b) add the chosen factor node to F, and update $\mathcal{R} \leftarrow \mathcal{R} \cup S_m$.

Clearly, the key step here is (2a), where the chosen set S_m ensures that the resulting factor graph is acyclic. This is because at any iteration, for the added factor node to form a cycle, it must connect at least two nodes in $\mathcal{R}$. However the added set always contains only one node

$\mathcal{R}$ and all others from $\mathcal{N} \setminus \mathcal{R}$. Define $N(i)$ as the set of neighboring nodes of node i, i.e., $N(i) = \{j : (i,j) \in \mathcal{E}\}$. In order to construct the connected subgraph of step (2a), it suffices to start at any node in the set $\{k : k \in \bigcup_{i \in \mathcal{R}} N(i), k \notin \mathcal{R}\}$ and traverse the subgraph $\mathcal{N} \setminus \mathcal{R}$ for $K-1$ steps. Overall, at most N possible graph-traversals may be required at each iteration, so the overall algorithm runs in time $O(NM)$.

The clusters C_j can also be constructed in a similar fashion, except that the nodes C_j added in step (2a) should be of the form $\{(i, C_{j_i}), i \in \mathcal{R}, C_{j_i} \subset N(i) \bigcap (\mathcal{N} \setminus \mathcal{R})\}$.

5.B Proofs Required for Section 5.3

Proof of Lemma 5.1

Given $\mathbf{x}$, the estimate $\hat{\mathbf{x}}$ depends on the mixing matrix $\mathbf{A}$, whose non-zero entries are chosen in an i.i.d. manner from $\mathbb{F}_Q$. The conditional probability of error can therefore be bounded as follows,

$$\text{Pr}(D_H(\hat{\mathbf{x}}, \mathbf{x}) \geq d | \mathbf{x}) \leq \text{Pr}\Big(\mathbf{A} \in \{\mathbf{A} : \exists \mathbf{z}, D_H(\mathbf{z}, \mathbf{x}) \geq d, \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{z}, p(\mathbf{z}) \geq p(\mathbf{x})\} | \, \mathbf{x}\Big) \tag{5.33a}$$

$$= \text{Pr}\left(\mathbf{A} \in \bigcup_{\substack{\mathbf{z} \in \mathbb{F}_Q^N, D_H(\mathbf{z},\mathbf{x}) \geq d \\ p(\mathbf{z}) \geq p(\mathbf{x})}} \{\mathbf{A} : \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{z}\} | \, \mathbf{x}\right) \tag{5.33b}$$

$$\leq \sum_{\substack{\mathbf{z} \in \mathbb{F}_Q^N, D_H(\mathbf{z},\mathbf{x}) \geq d \\ p(\mathbf{z}) \geq p(\mathbf{x})}} \text{Pr}(\mathbf{A} \in \{\mathbf{A} : \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{z}\} | \, \mathbf{x}) \tag{5.33c}$$

$$\leq \sum_{\substack{\mathbf{z} \in \mathbb{F}_Q^N, D_H(\mathbf{z},\mathbf{x}) \geq d \\ p(\mathbf{z}) \geq p(\mathbf{x})}} \prod_{m=1}^{M} \text{Pr}(\mathbf{a}_m^T \in \{\mathbf{a}_m^T : \mathbf{a}_m^T \mathbf{x} = \mathbf{a}_m^T \mathbf{z}\} | \, \mathbf{x}) \tag{5.33d}$$

The first inequality arises since the set in right hand side of (5.33a) also counts the case $p(\mathbf{z}) = p(\mathbf{x})$ (with $D_H(\mathbf{z}, \mathbf{x}) \geq d$) as an error. Such a situation may arise if $\hat{\mathbf{x}}$ is not unique. The inequality in (5.33c) is the union bound, while (5.33d) follows from the fact that the rows of $\mathbf{A}$ (denoted by $\mathbf{a}_m^T$) are independent.

Next recall that for $\mathbf{a}_m^T$, only the entries corresponding to the nodes in C_m are non-zero, and are chosen i.i.d. from $\mathbb{F}_Q$. Thus, given two vectors $\mathbf{x}$ and $\mathbf{z}$, it holds that [47],

$$\Pr(\mathbf{a}_m^T \in \{\mathbf{a}_m^T : \mathbf{a}_m^T\mathbf{x} = \mathbf{a}_m^T\mathbf{z}\}|\ \mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x}_{C_m} = \mathbf{z}_{C_m} \\ \frac{1}{Q} & \text{if } \mathbf{x}_{C_m} \neq \mathbf{z}_{C_m} \end{cases} \qquad 1 \leq m \leq M. \tag{5.34}$$

Since there are $MD_H(\mathbf{z}, \mathbf{x})$ clusters such that $\mathbf{x}_{C_m} \neq \mathbf{z}_{C_m}$,

$$\Pr(\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{z}|\mathbf{x}) = Q^{-MD_H(\mathbf{z},\mathbf{x})} \leq Q^{-dM}. \tag{5.35}$$

where the last inequality follows from the fact that $D_H(\mathbf{z}, \mathbf{x}) \geq d$.

Proof of Lemma 5.2

Observe that the pmf $p(\mathbf{x})$ in (5.18) can be expressed as

$$p(\mathbf{x}) = \frac{1}{Q} 2^{\sum_{q,\boldsymbol{i}} \ell_{q,\boldsymbol{i}}(\mathbf{x}) \log p_{q,\boldsymbol{i}}} \tag{5.36}$$

$$= \frac{1}{Q} 2^{-ME_{\mathbf{L}(x)}}. \tag{5.37}$$

Here, the exponent $E_{\mathbf{L}(x)}$ can be written as

$$E_{\mathbf{L}(x)} = -\frac{1}{M} \sum_{q=1}^{Q} \sum_{\boldsymbol{i} \in I_q} \ell_{q,\boldsymbol{i}}(\mathbf{x}) \log p_{q,\boldsymbol{i}} \tag{5.38a}$$

$$= -\sum_{q=1}^{Q} \frac{\ell_q(\mathbf{x})}{M} \sum_{\boldsymbol{i} \in I_q} \frac{\ell_{q,\boldsymbol{i}}(\mathbf{x})}{\ell_q(\mathbf{x})} \log p_{q,\boldsymbol{i}} \tag{5.38b}$$

$$= \sum_{q=1}^{Q} \frac{\ell_q(\mathbf{x})}{M} H(\varphi_q(\mathbf{x})) + \sum_{q=1}^{Q} \frac{\ell_q(\mathbf{x})}{M} D(\varphi_q(\mathbf{x}) \| p_q) \tag{5.38c}$$

$$=: H_{\mathbf{L}(\mathbf{x})} + D_{\mathbf{L}(\mathbf{x}),p} \tag{5.38d}$$

which is the exponent in (5.21) since $H(.)$ and $D(.)$ are defined as

$$H(\varphi(q)) := \sum_{i \in I_q} \frac{\ell_{q,\boldsymbol{i}}(\mathbf{x})}{\ell_q(\mathbf{x})} \log\left(\frac{\ell_q(\mathbf{x})}{\ell_{q,\boldsymbol{i}}(\mathbf{x})}\right), \tag{5.39}$$

$$D(\varphi(q) \| p_q) := \sum_{i \in I_q} \frac{\ell_{q,\boldsymbol{i}}(\mathbf{x})}{\ell_q(\mathbf{x})} \log\left(\frac{\ell_{q,\boldsymbol{i}}(\mathbf{x})}{\ell_q(\mathbf{x}) p_{q,\boldsymbol{i}}}\right). \tag{5.40}$$

Bound on $|\mathcal{T}(\mathbf{\Lambda})|$

Given a type $\mathbf{\Lambda}$, consider $p_{\mathbf{\Lambda}}(\mathbf{z})$ which also factors according to (5.18), but with transition probabilities specified by $[\mathbf{\Lambda}]_{q,i}$. In this case, $p_{\mathbf{\Lambda}}(\mathbf{z}) = \frac{1}{Q}2^{-MH_{\mathbf{\Lambda}}}$ since the term involving the KL-divergence in (5.21) vanishes. Drawing vectors $\mathbf{z}$ from this pmf, it follows that

$$\begin{aligned} 1 &\geq \sum_{\mathbf{z}\in\mathcal{T}(\mathbf{\Lambda})} p_{\mathbf{\Lambda}}(\mathbf{z}) && (5.41)\\ &\geq \frac{1}{Q}\sum_{\mathbf{z}\in\mathcal{T}(\mathbf{\Lambda})} 2^{-MH_{\mathbf{\Lambda}}} && (5.42)\\ &\geq \frac{1}{Q}|\mathcal{T}(\mathbf{\Lambda})|2^{-MH_{\mathbf{\Lambda}}} \end{aligned}$$

which yields the bound $|\mathcal{T}(\mathbf{\Lambda})| \leq Q2^{MH_{\mathbf{\Lambda}}}$.

Proof of Proposition 5.3

Start from the error expression in (5.14) and enumerate $\mathbf{x}$ using all possible *tree-based* types as in (5.22)

$$P_e = \sum_{\mathbf{\Lambda}_T\in\mathcal{L}_T}\sum_{\mathbf{x}\in\mathcal{T}(\mathbf{\Lambda}_T)} P_{e|\mathbf{\Lambda}_T}\frac{1}{Z}2^{-|\mathcal{E}_T|(H_{\mathbf{\Lambda}_T}+D_{\mathbf{\Lambda}_T,p})}2^{-|\bar{\mathcal{E}}_T|(H_{\bar{\mathbf{\Lambda}}_T}+D_{\bar{\mathbf{\Lambda}}_T,p})}. \tag{5.43}$$

Emulating the steps in Appendix 5.B, it can be shown that the number of vectors $\mathbf{x}$ of type $\mathbf{\Lambda}_T$ is bounded as $|\mathcal{T}(\mathbf{\Lambda}_T)| \leq Q2^{|\mathcal{E}_T|H_{\mathbf{\Lambda}_T}}$. It can be likewise shown that the number of vectors of the overall type $\mathbf{\Lambda} := (\mathbf{\Lambda}_T, \bar{\mathbf{\Lambda}}_T)$ is bounded as

$$|\mathcal{T}(\mathbf{\Lambda})| \leq Z_{\mathbf{\Lambda}}2^{|\mathcal{E}_T|H_{\mathbf{\Lambda}_T}}2^{|\bar{\mathcal{E}}_T|H_{\bar{\mathbf{\Lambda}}_T}} \tag{5.44}$$

where $Z_{\mathbf{\Lambda}} := \sum_{\mathbf{x}} 2^{-|\mathcal{E}|(H_{\mathbf{\Omega}(\mathbf{x})}+D_{\mathbf{\Omega}(\mathbf{x}),\mathbf{\Lambda}})}$. Since any set of edges $\mathcal{E}$ containing $\mathcal{E}_T$ allows for more vectors $\mathbf{x}$ of the same type than $\mathcal{E}_T$, then $|\mathcal{T}(\mathbf{\Lambda}_T)| \leq |\mathcal{T}(\mathbf{\Lambda})|$, and the following bound holds

$$\frac{1}{Z_{\mathbf{\Lambda}}}2^{-|\bar{\mathcal{E}}_T|H_{\bar{\mathbf{\Lambda}}_T}} \leq \frac{1}{|\mathcal{T}(\mathbf{\Lambda}_T)|}2^{|\mathcal{E}_T|H_{\mathbf{\Lambda}_T}}. \tag{5.45}$$

Substituting (5.45) into (5.43) yields

$$P_e \leq \sum_{\mathbf{\Lambda}_T \in \mathcal{L}_T} \frac{1}{|\mathcal{T}(\mathbf{\Lambda}_T)|} \sum_{\mathbf{x} \in \mathcal{T}(\mathbf{\Lambda}_T)} P_{e|\mathbf{\Lambda}_T} \frac{Z_{\mathbf{\Lambda}}}{Z} 2^{-(|\mathcal{E}_T| D_{\mathbf{\Lambda}_T,p} + |\bar{\mathcal{E}}_T| D_{\bar{\mathbf{\Lambda}}_T,p})} \tag{5.46}$$

$$\leq \sum_{\mathbf{\Lambda}_T \in \mathcal{L}_T} P_{e|\mathbf{\Lambda}_T} \frac{Z_{\mathbf{\Lambda}}}{Z} 2^{-(|\mathcal{E}_T| D_{\mathbf{\Lambda}_T,p} + |\bar{\mathcal{E}}_T| D_{\bar{\mathbf{\Lambda}}_T,p})}. \tag{5.47}$$

Clearly, $\frac{Z_{\mathbf{\Lambda}}}{Z} 2^{-(|\mathcal{E}_T| D_{\mathbf{\Lambda}_T,p} + |\bar{\mathcal{E}}_T| D_{\mathbf{\Lambda}_T,p})}$ equals one when $\mathbf{\Lambda} = p$, and decays exponentially when $\mathbf{\Lambda} \neq p$.

Chapter 6

Dynamic Network Delay Cartography

Path delays in IP networks are important metrics, required by network operators for assessment, planning, and fault diagnosis. Monitoring delays of all source-destination pairs in a large network is however challenging and wasteful of resources. The present chapter advocates a spatio-temporal Kalman filtering approach to construct network-wide delay maps using measurements on only a few paths. The proposed network cartography framework allows efficient tracking and prediction of delays by relying on both topological as well as historical data. Optimal paths for delay measurement are selected in an online fashion by leveraging the notion of submodularity. The resulting predictor is optimal in the class of linear predictors, and outperforms competing alternatives on real-world datasets.

This chapter is organized as follows. Section 3.1 introduces the model and the problem statement. Section 6.2 deals with the Kriged Kalman Filter (KKF) approach, while Section 6.2.1 describes techniques for estimating the relevant parameters. Finally, empirical validation of KKF and comparisons with the Kriging approach of [29] are provided in Section 6.4.

Notation. Lower case symbols with indices, such as y_p, represent scalar variables. These variables, when stacked over their indices are denoted through their bold-faced versions $\mathbf{y}$. Bold-faced upper case symbols ($\mathbf{S}$) represent matrices. Regular upper case symbols (S) represent constant scalars, and typically stand for the cardinality of the set represented by corresponding calligraphic upper case symbol ($\mathcal{S}$). Identity matrix of size $P \times P$ is denoted by $\mathbf{I}_P$, , and its columns by $\mathbf{e}_1$, $\mathbf{e}_2$, . . ., $\mathbf{e}_P$. Matrix $\mathbf{C}_\mathbf{y}$ denotes the covariance matrix of the vector $\mathbf{y}$.

6.1 Modeling and Problem Statement

Consider an IP network modeled by a connected digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ denoting the set of nodes (devices, servers, or routers), and $\mathcal{E}$, the communication links. The issue is to monitor path delays on a set of multi-hop paths $\mathcal{P}$ that connect the $P := |\mathcal{P}|$ source-destination pairs. Latency measured on path $p \in \mathcal{P}$ at time t is denoted by $y_p(t)$, and all such network-wide delays are collected in the vector $\mathbf{y}(t)$. At any time t however, delay can only be measured on a subset of paths $\mathcal{S}(t) \subset \mathcal{P}$, which is represented by $\mathbf{y}_s(t)$. Based on such partial current and past measurements $\mathcal{H}(t) := \{\mathbf{y}_s(\tau)\}_{\tau=1}^{t}$, the goal is to predict the remaining path delays $\mathbf{y}_{\bar{s}}(t) := \{y_p(t)\}_{p \in \mathcal{P} \setminus \mathcal{S}(t)}$ for each t.

The per-path end-to-end delay $y_p(t)$ comprises of several independent components corresponding to contributions from each intermediate link and router. Of these, the queuing delay $\chi_p(t)$ is the time spent by the packets waiting in the queues of intermediate buffers, and depends on the traffic volumes in competing links. Network traffic is not only correlated spatio-temporally, but also exhibits periodic behavior, random fluctuations, and occasional bursts [87]. These effects motivate the following random-walk model for the latent vector of queuing delays $\boldsymbol{\chi}(t)$,

$$\boldsymbol{\chi}(t) = \boldsymbol{\chi}(t-1) + \boldsymbol{\eta}(t) \tag{6.1}$$

where $\boldsymbol{\eta}(t)$ denotes state noise with zero mean and covariance matrix $\mathbf{C}_{\boldsymbol{\eta}} := \mathbb{E}\left[\boldsymbol{\eta}(t)\boldsymbol{\eta}^T(t)\right]$.

Other components of the path delay, combined in the term $\nu_p(t)$, include the propagation, processing and transmission delays, which are temporally uncorrelated (see e.g., [17] for details). The delays $\nu_p(t)$ are still zero mean, spatially correlated across paths, and the covariance matrix of the compacted vector $\boldsymbol{\nu}(t)$ is given by $\mathbf{C}_{\boldsymbol{\nu}}$. Finally, the measurement of path delays using software tools such as `ping` itself introduces errors $\epsilon_p(t)$, which are assumed zero mean, uncorrelated over time and across paths, with covariance $\sigma^2 := \mathbb{E}\left[\epsilon_p(t)\epsilon_p^T(t)\right]$.

The measured delays are expressed as

$$y_p(t) = \chi_p(t) + \nu_p(t) + \epsilon_p(t) \qquad\qquad p \in \mathcal{S}(t).$$

Letting $\mathbf{S}(t)$ denote the $|\mathcal{S}(t)| \times P$ selection matrix with 0-1 entries that contains the p-th row of $\mathbf{I}_P$ if $p \in \mathcal{S}(t)$, the measurement equation can be compactly written as

$$\mathbf{y}_s(t) = \mathbf{S}(t)\boldsymbol{\chi}(t) + \boldsymbol{\nu}_s(t) + \boldsymbol{\epsilon}_s(t) \tag{6.2}$$

where the vector $\boldsymbol{\epsilon}_s(t)$ collects the measurement errors on paths $p \in \mathcal{S}(t)$, and $\boldsymbol{\nu}_s(t) := \mathbf{S}(t)\boldsymbol{\nu}(t)$.

The next section describes a KKF approach for tracking and predicting the end-to-end delays $\mathbf{y}_{\bar{s}}(t)$, by utilizing the state-space model described by (6.1) and (6.2).

6.2 Dynamic Network Kriging

The spatio-temporal model in (6.1)–(6.2) is widely employed in geostatistics and environmental science, where $\boldsymbol{\chi}(t)$ is generally referred to as trend, and $\boldsymbol{\nu}(t)$ captures random fluctuations around $\boldsymbol{\chi}(t)$; see e.g., [141, Ch. 4], [104, 166]. Recently, a similar modeling approach was employed by [82] to describe the dynamics of wireless propagation channels, and in [33] for spatio-temporal random field estimation. Given only first- and second-order moments of $\boldsymbol{\eta}(t)$, $\boldsymbol{\epsilon}_s(t)$, and $\boldsymbol{\nu}(t)$, this section derives the best linear predictor for $\mathbf{y}_{\bar{s}}(t)$.

Suppose first that the queuing delay vector $\boldsymbol{\chi}(t)$ is known, and let $\bar{\mathbf{S}}(t)$ denote an $|\bar{\mathcal{S}}(t)| \times P$ matrix containing the p-th row of $\mathbf{I}_P$ if $p \in \bar{\mathbf{S}}(t)$; that is, $\bar{\mathbf{S}}(t)$ is a path selection matrix which returns quantities pertaining to paths in $\bar{\mathbf{S}}(t)$. Then, the linear minimum mean-square error (LMMSE) estimator (denoted by $\mathbb{E}^*[.]$) for $\boldsymbol{\nu}_{\bar{s}}(t)$ is given by (see, e.g. [4])

$$\mathbb{E}^*\left[\boldsymbol{\nu}_{\bar{s}}(t)|\boldsymbol{\chi}(t)\right] = \bar{\mathbf{S}}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)\left(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right)^{-1}\left[\mathbf{y}_s(t) - \mathbf{S}(t)\boldsymbol{\chi}(t)\right] \tag{6.3}$$

and is commonly referred to as kriging [36]. In practice however, the trend $\boldsymbol{\chi}(t)$ has to be estimated from the data. In the so-termed universal kriging predictor [141], $\boldsymbol{\chi}(t)$ is estimated using the generalized least-squares (GLS) criterion, where $\boldsymbol{\nu}_s(t)$ is treated as noise (lumped together with $\boldsymbol{\epsilon}_s(t)$). The prediction for $\boldsymbol{\nu}_{\bar{s}}(t)$ is then obtained by replacing $\boldsymbol{\chi}(t)$ in (6.3) with its estimate. This approach was proposed for network delay prediction in [29], and was referred to as network kriging. However, since the trend is estimated independently using GLS per time slot, its temporal dynamics present in (6.1) are not exploited.

From the spatio-temporal model set forth in Section 6.1, it is clear that estimating the trend $\boldsymbol{\chi}(t)$ can benefit from processing both present and past measurements jointly. Towards this end, the Kalman filtering (KF) machinery offers a viable option for tracking the evolution of $\boldsymbol{\chi}(t)$ from the set of historical data $\mathcal{H}(t)$. At each time t, the KF finds the LMMSE estimate $\hat{\boldsymbol{\chi}}(t) := \mathbb{E}^*\left[\boldsymbol{\chi}(t)|\mathcal{H}(t)\right]$, and its error covariance matrix $\mathbf{M}(t) := \mathbb{E}\left[(\boldsymbol{\chi}(t)-\hat{\boldsymbol{\chi}}(t))(\boldsymbol{\chi}(t)-\hat{\boldsymbol{\chi}}(t))^T\right]$ using the following set of recursions (see e.g., [4, Ch. 3])

$$\hat{\boldsymbol{\chi}}(t) = \hat{\boldsymbol{\chi}}(t-1) + \mathbf{K}(t)(\mathbf{y}_s(t) - \mathbf{S}(t)\hat{\boldsymbol{\chi}}(t-1)) \tag{6.4a}$$

$$\mathbf{M}(t) = (\mathbf{I}_P - \mathbf{K}(t)\mathbf{S}(t))(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}}) \tag{6.4b}$$

where the so-termed Kalman gain $\mathbf{K}(t)$ is given by

$$\mathbf{K}(t) := (\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}})\mathbf{S}^T(t)\left[\mathbf{S}(t)(\mathbf{C}_{\boldsymbol{\nu}} + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{M}(t-1))\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right]^{-1}. \tag{6.5}$$

Once $\hat{\boldsymbol{\chi}}(t)$ has been estimated via KF, $\boldsymbol{\nu}_{\bar{s}}(t)$ can be readily obtained via kriging as in (6.3), yielding the predictor

$$\begin{aligned}\hat{\mathbf{y}}_{\bar{s}}(t) = \hat{\boldsymbol{\chi}}(t) &+ \bar{\mathbf{S}}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)\left(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right)^{-1} \\ &\times\left[\mathbf{y}_s(t) - \mathbf{S}(t)\hat{\boldsymbol{\chi}}(t)\right].\end{aligned} \tag{6.6}$$

The predictor in (6.6) constitutes what is also referred to as the kriged Kalman filter [104,166]. The LMMSE framework employed here yields the best linear predictor even for non-Gaussian distributed noise. The prediction error of the KKF is characterized in the following proposition, whose proof is provided in Appendix 6.A.

Proposition 6.1. *The prediction error covariance matrix at time t is given by*

$$\mathbf{M}^{\mathbf{y}}_{\bar{s}}(t) := \mathbb{E}\{(\mathbf{y}_{\bar{s}}(t) - \hat{\mathbf{y}}_{\bar{s}}(t))(\mathbf{y}_{\bar{s}}(t) - \hat{\mathbf{y}}_{\bar{s}}(t))^T\} \tag{6.7a}$$

$$= \sigma^2\mathbf{I}_{\bar{S}} + \bar{\mathbf{S}}(t)\left[(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\nu}} + \mathbf{C}_{\boldsymbol{\eta}})^{-1} + \frac{1}{\sigma^2}\mathbf{S}^T(t)\mathbf{S}(t)\right]^{-1}\bar{\mathbf{S}}^T(t). \tag{6.7b}$$

Having a closed-form expression for the prediction error will come handy for selecting the matrix $\mathbf{S}(t)$, as shown later in Section 6.3.

The KF step also allows τ-step prediction for $\tau \geq 1$, which is given by $\hat{\mathbf{y}}(t+\tau) = \hat{\boldsymbol{\chi}}(t)$, since the kriging term is temporally white. In the present context, this can be useful in preemptive routing and congestion control algorithms, as well as for extrapolating missing measurements. In the latter case, the covariance matrix is updated simply as $\mathbf{M}(t) = \mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}}$. Before concluding the description of the KKF, the following remarks are due.

Remark 6.1. The random walk model adopted in (6.1) may result in an unstable filter. Operationally, if the KKF is unstable, an incorrect initialization of $\mathbf{M}(0)$ or $\boldsymbol{\chi}(0)$ may result in poor prediction performance even as $t \to \infty$. This can be remedied by adopting a damped model $\boldsymbol{\chi}(t) = \kappa\boldsymbol{\chi}(t-1) + \boldsymbol{\eta}(t)$ with $\kappa < 1$. The results presented in this chapter also generalize to the damped case. The random walk model is nevertheless used here since no instability issues were observed in the two data sets considered in Section 6.4.

Remark 6.2. A distributed implementation of the KKF may be desirable for enhancing the robustness and scalability of delay monitoring. In large-scale networks, a distributed algorithm also mitigates the message passing overhead required to collect all measurements at a fusion center. If the model covariances $\mathbf{C}_{\boldsymbol{\nu}}$ and $\mathbf{C}_{\boldsymbol{\eta}}$ are globally known, and the selection matrix $\mathbf{S}(t)$ is constant for all t, a distributed implementation of (6.4) can be derived along the lines of [42]. On the other hand, if each node of the network has partial knowledge of $\mathbf{C}_{\boldsymbol{\nu}}$, $\mathbf{C}_{\boldsymbol{\eta}}$ and $\mathbf{S}(t)$, the algorithm developed in [33] can be appropriately tailored to the problem at hand.

6.2.1 Estimating Model Parameters

The LMMSE-optimal dynamic kriging framework described in Section 6.2 requires knowledge of model covariance matrices $\mathbf{C}_{\boldsymbol{\nu}}$, $\sigma^2\mathbf{I}_S$, and $\mathbf{C}_{\boldsymbol{\eta}}$, to operate. Of these, σ^2 depends on the precision offered by the measurement software, and can be safely assumed known a priori.

The structure of $\mathbf{C}_{\boldsymbol{\nu}}$ is motivated by the modeling assumptions and utilizes topological information. Intuitively, propagation, transmission, and processing delays over paths $p, q \in \mathcal{P}$ should be highly correlated if these paths share many links. This relationship can be modeled by utilizing the Gramian matrix $\mathbf{G} := \mathbf{R}\mathbf{R}^T$, where $\mathbf{R}$ is the $P \times |\mathcal{E}|$ path-link routing matrix; that is, the (p, l)th element of $\mathbf{R}$ is 1 if path $p \in \mathcal{P}$ traverses link $l \in \mathcal{E}$, and 0 otherwise. Each off-diagonal entry (p, q) of $\mathbf{G}$ represents the number of links common to the paths $p, q \in \mathcal{P}$. On the other hand, the elements on the main diagonal of $\mathbf{G}$ count the number of constituent links per path. The covariance matrix of $\boldsymbol{\nu}(t)$ can therefore be modeled as $\mathbf{C}_{\boldsymbol{\nu}} = \gamma\mathbf{G}$. A similar model for $\mathbf{C}_{\boldsymbol{\nu}}$ was adopted by [29], where it was motivated from the property that path delays are sum of link delays, that is, $\boldsymbol{\nu}(t) = \mathbf{R}\mathbf{x}(t)$, where vector $\mathbf{x}(t)$ collects the link delays. Under this assumption, it holds that $\mathbf{C}_{\boldsymbol{\nu}} = \gamma\mathbf{G}$ if the link delays are uncorrelated across links, and have covariance matrix $\gamma\mathbf{I}_{|\mathcal{E}|}$.

For the remaining parameters, namely γ and $\mathbf{C}_{\boldsymbol{\eta}}$, an empirical approach is described next. It entails a training phase, and a set of measurements $\{\mathbf{y}_s(t)\}_{t=1}^{t_L}$ collected at time slots $t = 1, \ldots, t_L$. During the KKF operation, $t_L - 1$ time slots can be periodically devoted to updating model covariances, while predicting the networks-wide delays $\mathbf{y}_{\bar{s}}(t)$ for $t = 1, \ldots, t_L$. Let $\widehat{\mathbf{C}}_{\boldsymbol{\nu}}(t) := \hat{\gamma}(t)\mathbf{G}$ and $\widehat{\mathbf{C}}_{\boldsymbol{\eta}}(t)$ denote the estimates of $\mathbf{C}_{\boldsymbol{\nu}}$ and $\mathbf{C}_{\boldsymbol{\eta}}$, respectively, at time t. Estimating the covariance matrix of the state noise is well-known to be a challenging task, primarily because $\boldsymbol{\chi}(t)$ and $\boldsymbol{\chi}(t-1)$ are not directly observable. Furthermore, methods such as those in [107] are not applicable in the present context, as they require the KF to be time-invariant and stationary. As shown in [111], a viable means of estimating $\mathbf{C}_{\boldsymbol{\eta}}$ from $\{\mathbf{y}_s(t)\}_{t=1}^{t_L}$ relies on approximating the noise $\boldsymbol{\eta}(t)$ as $\boldsymbol{q}(t) := \hat{\boldsymbol{\chi}}(t) - \hat{\boldsymbol{\chi}}(t-1)$. Then, upon noticing that the resultant process $\{\boldsymbol{q}(\tau)\}$ is temporally-white, the sample mean and covariance of $\boldsymbol{q}$ can be obtained as

$$\hat{\boldsymbol{m}}_{\boldsymbol{q}}(t_L) = \frac{1}{t_L - 1}\sum_{t=2}^{t_L}\boldsymbol{q}(t) \tag{6.8}$$

$$\widehat{\mathbf{C}}_{\boldsymbol{q}}(t_L) = \frac{1}{t_L - 2}\sum_{t=2}^{t_L}(\boldsymbol{q}(t) - \hat{\boldsymbol{m}}_{\boldsymbol{q}}(t))(\boldsymbol{q}(t) - \hat{\boldsymbol{m}}_{\boldsymbol{q}}(t))^T. \tag{6.9}$$

Using (6.9), and exploiting the equality $\mathbb{E}\{\widehat{\mathbf{C}}_{\boldsymbol{q}}\} = (t_L - 1)^{-1}\sum_t(\mathbf{M}(t-1) - \mathbf{M}(t)) + \mathbf{C}_{\boldsymbol{\eta}}$, it follows that an unbiased estimate of $\mathbf{C}_\eta$ can be obtained as

$$\widehat{\mathbf{C}}_\eta(t_L) = \widehat{\mathbf{C}}_{\boldsymbol{q}}(t_L) + \frac{1}{t_L - 1}\sum_{t=2}^{t_L}\Big(\mathbf{M}(t) - \mathbf{M}(t-1)\Big). \tag{6.10}$$

Finally, in order to obtain $\hat{\gamma}$, consider the innovations at time t as $\iota_p(t) := y_p(t) - \hat{\chi}_p(t-1)$, and notice that if the model covariances are correct, then $\iota_p(t)$ is temporally white and zero-mean [107]. Indeed, it is possible to show that $\mathbb{E}\left[\iota_p(t)\iota_q(t)\right] = \left[\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}}\right]_{pq} + \sigma^2$ for any $p, q \in \mathcal{S}(t)$ [111]. Further, let $\mathcal{T}_{pq} := \{t | 1 \leq t \leq t_L, p, q \in \mathcal{S}(t)\}$ be the set of time slots for which paths p and q are both measured. Then, the sample covariance between $\iota_p(t)$ and $\iota_q(t)$ is given by $\hat{C}_{pq} := |\mathcal{T}_{pq}|^{-1}\sum_{t\in\mathcal{T}_{pq}}\iota_p(t)\iota_q(t)$ for all pairs $p, q \in \mathcal{P}$. Given $\mathbf{M}(t-1)$ and σ^2, this observation yields the following estimate

$$\left[\widehat{\mathbf{C}}_{\boldsymbol{\nu}}(t)\right]_{pq} = \frac{1}{|\mathcal{T}_{pq}|}\sum_{t\in\mathcal{T}_{pq}}\iota_p(t)\iota_q(t) - \sigma^2 - [\mathbf{M}(t-1) + \widehat{\mathbf{C}}_{\boldsymbol{\eta}}(t)]_{pq}. \tag{6.11}$$

Indeed, entries of $\widehat{\mathbf{C}}_{\boldsymbol{\nu}}(t)$ can be updated recursively using $\widehat{\mathbf{C}}_{\boldsymbol{\nu}}(t-1)$ in (6.11). At each time, only a few entries are updated, depending on which paths are observed.

Finally, $\hat{\gamma}(t)$ can be obtained by fitting $\widehat{\mathbf{C}}_{\boldsymbol{\nu}}(t)$ to $\gamma\mathbf{G}$ in the least-squares sense, which yields

$$\hat{\gamma}(t_L) = \frac{\sum_{p,q\in\mathcal{P}}[\mathbf{G}]_{pq}[\widehat{\mathbf{C}}_{\boldsymbol{\nu}}(t_L)]_{pq}}{\|\mathbf{G}\|_F^2}. \tag{6.12}$$

6.3 Online Experiment Design

This section considers the problem of optimally choosing the set of paths $\mathcal{S}(t)$ (equivalently, the matrix $\mathbf{S}(t)$) so as to minimize the prediction error. To begin with, a simple case is considered where the set $\mathcal{S}(t)$ is allowed to contain any S paths. Operational requirements may however impose further constraints on $\mathcal{S}(t)$, and these are discussed later.

The prediction error can be characterized by using a scalar function of $\mathbf{M}_{\bar{s}}^{\mathbf{y}}(t)$; see e.g., [5]. To this end, the so called D-optimal design is considered, where the goal is to minimize the function $f(\mathcal{S}(t)) := \log\det(\mathbf{M}_{\bar{s}}^{\mathbf{y}}(t))$. The paths selected at time t are therefore given by the solution of the following optimization problem

$$\mathcal{S}^*(t) = \arg\min_{\mathcal{S}\in\mathcal{P}} f(\mathcal{S}) \tag{6.13}$$

$$\text{s. t.} \qquad |\mathcal{S}| = S. \tag{6.14}$$

Clearly, tackling (6.13) incurs combinatorial complexity and is challenging to solve exactly, even for moderate-size networks. Indeed, (6.13) is an example of the so called subset selection problem, which is NP-complete in general; see e.g., [43] and references therein.

Interestingly, it is possible to solve (6.13) approximately by utilizing the notion of *submodularity*. Consider a function $g(\mathcal{S})$, which takes as input sets $\mathcal{S}\subset\mathcal{P}$. Given a set $\mathcal{A}\in\mathcal{P}$ and an element $p\in\mathcal{P}\setminus\mathcal{A}$, the increment function is defined as $\delta_{\mathcal{A}}^g(p) := g(\mathcal{A}\cup\{p\}) - g(\mathcal{A})$. Function $g(\cdot)$ is submodular if its increments are monotonically decreasing, meaning $\delta_{\mathcal{A}}^g(p) \geq \delta_{\mathcal{B}}^g(p)$ for all $\mathcal{A}\subset\mathcal{B}\in\mathcal{P}$. Likewise, $g(\cdot)$ is supermodular iff $\delta_{\mathcal{A}}^g(p) \leq \delta_{\mathcal{B}}^g(p)$ for all $\mathcal{A}\subset\mathcal{B}\in\mathcal{P}$. In the present case, the following proposition holds.

Proposition 6.2. *The function $f(\mathcal{S})$ is monotonic and supermodular in $\mathcal{S}$.*

The proof of Proposition 6.2 is provided in Appendix 6.B, and relies on related results from [5].

An important implication of Proposition 6.2 is that a greedy forward selection algorithm can be developed to solve (6.13) approximately [114]. Upon defining the shifted function $h(\mathcal{S}) := f(\mathcal{S}) - \log\det(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}} + \sigma^2\mathbf{I}_P)$, a result from [114] ensures that the solution of the greedy algorithm $\mathcal{S}^g(t)$ satisfies the inequality

$$h(\mathcal{S}^g(t)) \leq \left(1 - \frac{1}{e}\right) h(\mathcal{S}^*(t)). \tag{6.15}$$

While performance of the greedy algorithm is usually much better in practice, this bound ensures that it does not break down for pathological inputs.

The greedy algorithm involves repeatedly performing the updates $\mathcal{S} \leftarrow \mathcal{S} \cup \arg\min_{p \notin \mathcal{S}} \delta_{\mathcal{S}}^f(p)$ until $|\mathcal{S}| = S$. This is useful in the present case, since the increments can be evaluated efficiently using determinant update rules. Specifically, the updates are given by

$$\delta_{\emptyset}^f(p) = -\log\left(1 + \left[\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}}\right]_{p,p}\right) \qquad \forall p \in \mathcal{P} \tag{6.16}$$

$$\delta_{\mathcal{S}}^f(p) = -\log\left(1 + \left[\left((\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}})^{-1} + \mathbf{S}^T\mathbf{S}\right)^{-1}\right]_{p,p}\right) \quad \forall p \in \mathcal{P} \setminus \mathcal{S}. \tag{6.17}$$

Further, each iteration requires a rank-one update to the matrix inverse in (6.17), which can also be performed efficiently. The full greedy approach is summarized in Algorithm 6.1, where $\boldsymbol{\Phi} := (\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}})/\sigma^2$. Algorithm 6.1 involves only basic operations, and it is easy to verify that its worst case complexity is $O(PS^3)$. Further, the final value of the matrix $\mathbf{V}$ evaluated in the last iteration (Algorithm 6.1, line 11) is exactly the inverse term required for evaluating the Kalman gain in (6.5). It is remarked that the operational complexity can be further reduced using lazy updates [109].

Next, consider a more practical scenario, where the software installed at each end-node can measure delays on all paths originating at that node. At any time t however, delays are measured from only N end-nodes. Let $\mathcal{V}_e$ denote the set of all end-nodes, and $\mathcal{P}_v$, the set of paths which have the node $v \in \mathcal{V}_e$ as their origin (likewise, $\mathcal{P}_{\mathcal{N}} := \bigcup_{v \in \mathcal{N}} \mathcal{P}_v$ for $\mathcal{N} \subset \mathcal{V}_e$). For any subset $\mathcal{N}$ (and its complement $\bar{\mathcal{N}} := \mathcal{V} \setminus \mathcal{N}$), define the selection matrix $\mathbf{N}$ ($\bar{\mathbf{N}}$) consisting of canonical vectors $\mathbf{e}_p^T$ as rows, for all $p \in \mathcal{P}_{\mathcal{N}}$ ($p \in \mathcal{P}_{\bar{\mathcal{N}}}$). Defining the cost

Algorithm 6.1: Greedy algorithm for solving (6.13)

```
Data: $\mathbf{\Phi}$, $S$
Result: $\mathcal{S}$
1  $s \leftarrow \arg\max_{1\le p\le P} [\mathbf{\Phi}]_{p,p}$
2  $\mathbf{V} \leftarrow \left[1/\left([\mathbf{\Phi}]_{s,s}+1\right)\right]$
3  $\mathcal{S} \leftarrow \{s\}$
4  for $k = 2$ to $S$ do
5      $\mathbf{w}_p \leftarrow \mathbf{\Phi}_{\mathcal{S},p}$ for all $p \in \mathcal{P} \setminus \mathcal{S}$
6      $s \leftarrow \arg\max_{p\notin\mathcal{S}} [\mathbf{\Phi}]_{p,p} - \mathbf{w}_p^T\mathbf{V}\mathbf{w}_p$
7      $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$
8      $d \leftarrow [\mathbf{\Phi}]_{s,s} - \mathbf{w}_s^T\mathbf{V}\mathbf{w}_s + 1$
9      $\mathbf{u} \leftarrow -\mathbf{V}\mathbf{w}_s$
10     $\mathbf{V} \leftarrow \begin{bmatrix} \mathbf{V}+\mathbf{u}\mathbf{u}^T/d & \mathbf{u}/d \\ \mathbf{u}^T/d & 1/d \end{bmatrix}$
11 end
```

function $f_n(\mathcal{N}) := f(\mathcal{P}_\mathcal{N})$, the online optimal design problem for this scenario is expressed as

$$\mathcal{N}^*(t) = \arg\min_{\mathcal{N}\subset\mathcal{V}_e} f_n(\mathcal{N}) \tag{6.18a}$$

$$\text{s. t.} \quad |\mathcal{N}| = N. \tag{6.18b}$$

It follows from the properties of submodular functions that the cost function $f_n(\mathcal{N})$ is also monotonic and supermodular in $\mathcal{N}$. In particular, observe that the increments $\delta^n_\mathcal{N}(v) = f_n(\mathcal{N}\cup\{v\}) - f_n(\mathcal{N}) = f(\mathcal{P}_\mathcal{N}\cup\mathcal{P}_v) - f(\mathcal{P}_\mathcal{N})$ for $v \notin \mathcal{N}$ satisfy the non-increasing property, i.e., $\delta^n_\mathcal{A}(v) \le \delta^n_\mathcal{B}(v)$ for all $\mathcal{A}\subset\mathcal{B}\subset\mathcal{V}_e$ and $v\notin\mathcal{B}$. A greedy algorithm similar to Algorithm 6.1 can therefore be developed to obtain an approximate solution with the same $(1-1/e)$ guarantee as in (6.15). Complexity of the greedy algorithm in this case would be however higher, since evaluating $\delta_\mathcal{N}(v)$ now requires rank-$|\mathcal{P}_v|$ updates in the determinant and inverses. Nevertheless, the algorithm would still be efficient as long as $|\mathcal{P}_v| \ll P$ for all $v\in\mathcal{V}_e$. In the special case when delay measurements are performed by only one node per time slot ($N=1$), the solution of (6.18a) is simply given by

$$\mathcal{N}^*(t) = \arg\min_{v\in\mathcal{V}_e} \log\det\left(\mathbf{I}_{|\mathcal{P}_v|} + \Big[\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}}\Big]_{vv}\right) \tag{6.19}$$

where $\Big[\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}}\Big]_{vv}$ is the $|\mathcal{P}_v|\times|\mathcal{P}_v|$ submatrix containing the rows and columns of $\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}}$ corresponding to the paths in $\mathcal{P}_v$.

In some networks, it may be relatively straightforward to install delay measurement software on every end-node, while allowing each end-node to measure delay on only one path per time slot. This amounts to replacing the budget-constraint (6.14) in (6.13) with

$$|\mathcal{S}\cap\mathcal{P}_v| = 1 \qquad\qquad \forall\, v\in\mathcal{V}_e. \tag{6.20}$$

Interestingly, constraints of this form can also be handled using the greedy approach by simply imposing (6.20) while searching for the best increment at every iteration. Specifically, the search space of path p [cf. Algorithm 6.1, line 7] now becomes $p \in \mathcal{P}\setminus\mathcal{P}_{\mathcal{N}}$, where $\mathcal{N} = \{v : \mathcal{S}\cap\mathcal{P}_v \neq \emptyset\}$. More general constraints of the form $|\mathcal{S}\cap\mathcal{P}_v| \leq S_v$ can similarly be incorporated. Constraints of this form are referred to as partition matroid constraints, under which the greedy algorithm provides an approximation ratio of $1/2$ [55].

6.4 Empirical Validation

Performance of the proposed network-wide latency prediction schemes is validated using two different datasets, which include delays measured on:

(a) Internet2 backbone network[1], a lightly loaded network that exhibits low delay variability; and,

(b) New Zealand Active Measurement Project (NZ-AMP)[2], a network deployed across several universities and ISPs in New Zealand, characterized by comparatively higher variability in delays.

[1][Online] `http://www.internet2.edu/network`
[2][Online] `http://erg.cs.waikato.ac.nz/amp`

Using the aforementioned datasets, the performance of KKF is also compared against that of competing alternatives in [29] and [30].

Before proceeding, a brief description of the nonlinear estimation technique in [30] is provided. The approach hinges on a sparse representation of the network-wide delays, and employs ℓ_1-norm minimization to recover the sparse basis coefficient vector. Specifically, the path delays adhere to the postulated linear model $\mathbf{y}(t) = \mathbf{H}\boldsymbol{\beta}(t)$, where $\|\boldsymbol{\beta}(t)\|_0 \ll P$, and the matrix $\mathbf{H} \in \mathbb{R}^{P\times P}$ is constructed using diffusion wavelets [31]. The diffusion matrix used for computing the wavelet basis is obtained by applying Sinkhorn balancing [154] to the matrix $\mathbf{W} \in \mathbb{R}^{P\times P}$, whose (p, q)-th element is defined as

$$[\mathbf{W}]_{p,q} = \frac{[\mathbf{G}]_{pq}}{[\mathbf{G}]_{pp} + [\mathbf{G}]_{qq} - [\mathbf{G}]_{pq}} \tag{6.21}$$

where $\mathbf{G}$ is the Gramian defined in Section 6.2.1. The overall algorithm amounts to solving the following minimization problem

$$\hat{\boldsymbol{\beta}}'(t) = \arg\min_{\boldsymbol{\beta}'} \|\boldsymbol{\beta}'\|_1 \tag{6.22a}$$

$$\text{s. t.} \quad \mathbf{y}_s(t) = \mathbf{S}(t)\mathbf{H}\mathbf{L}\boldsymbol{\beta}' \tag{6.22b}$$

where $\mathbf{L}$ is a diagonal matrix whose (n, n)-th entry is given by $[\mathbf{L}]_{n,n} = 2^k$, with $k \in \mathbb{N}$ denoting the scale corresponding to the diffusion wavelet coefficient β_n [30]. Subsequently, $\mathbf{y}_{\bar{s}}(t)$ is predicted as $\hat{\mathbf{y}}_{\bar{s}}(t) = \bar{\mathbf{S}}(t)\mathbf{H}\mathbf{L}\hat{\boldsymbol{\beta}}'(t)$.

Under the premise that delays change slowly with time, the described algorithm can be used to estimate $\mathbf{y}_{\bar{s}}(t)$ over a sequence of $\tau > 1$ contiguous time-steps jointly. In this case, problem (6.22) is solved by replacing $\mathbf{y}_s(t)$ with $\bar{\mathbf{y}}_s(t) := [\mathbf{y}_s^T(t-\tau+1), \mathbf{y}_s^T(t-\tau+2), \ldots, \mathbf{y}_s^T(t)]^T$, and by computing the $P\tau \times P\tau$ diffusion wavelet matrix based on $\mathbf{W}$ and temporal correlations as shown in [30]. Although this is a viable way to capture temporal correlations of delays, observe that it requires solving ℓ_1-norm minimization problems with $P\tau$ variables every τ time slots. This increase in complexity prohibits the use of a large value of τ, and the simulations here only report performance with $\tau = 5$. It is also worth mentioning that such a batch solution also does not compare favorably to a real-time implementation, such as that provided by the KKF where delay predictions become available every time new measurements arrive.

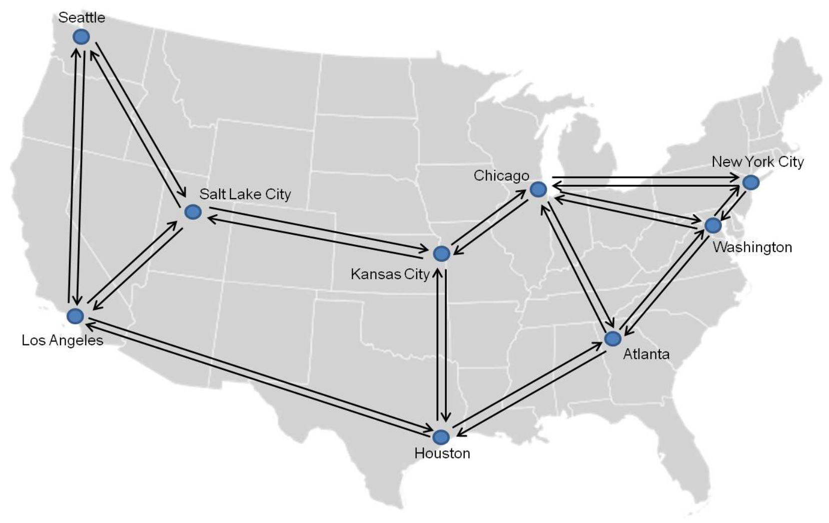


Figure 6.1: Internet2 IP backbone network.

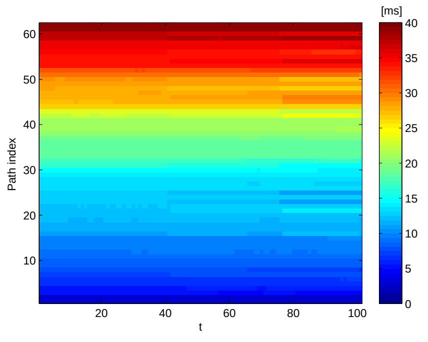
6.4.1 Internet2 Delay Data

The One Way Active Measurement Project (OWAMP) collects one way delays on the Internet2 backbone network[3]. The network has 9 end-nodes and 26 directional links as depicted in Figure 6.1. Delays are measured on the 72 paths among the end-nodes every minute. The data $\{\mathbf{y}(t)\}$ is collected over $t_P = 4500$ minutes (about three days) in July 2011.

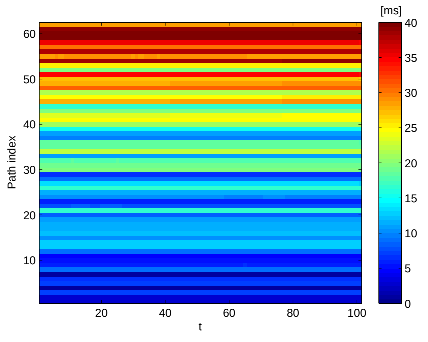
The model KKF covariances $\mathbf{C}_{\boldsymbol{\nu}}$ and $\mathbf{C}_{\boldsymbol{\eta}}$ are estimated using data from the initial 1,000 time slots. In this phase, 50 paths are randomly selected per time slot. The KKF is initialized by setting $\gamma = 1$, $\mathbf{C}_{\boldsymbol{\eta}} = \mathbf{C}_{\boldsymbol{\nu}}$, and run for 500 time slots. Next, $\hat{\gamma}(t)$ and $\widehat{\mathbf{C}}_{\eta}(t)$ are updated in an online fashion, as outlined in Section 6.2.1. The final values are obtained at the conclusion of the training phase at t = 1,000.

Pictorially, the performance of different algorithms can be assessed through delay maps shown in 6.2. Such maps can succinctly represent the network health, and are especially useful for networks which otherwise have low delay variability, such as the Internet2. The map in Figure 6.2(a) corresponds to the true delays, where maps (b), (c), and (d) depict the predicted values obtained from the network kriging, wavelet-based approach, and KKF respectively.

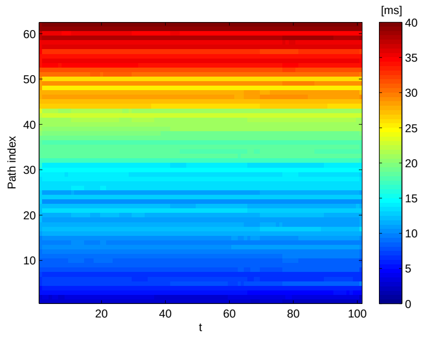
[3][Online] `http://ndb1.net.internet2.edu/cgi-bin/owamp.cgi`



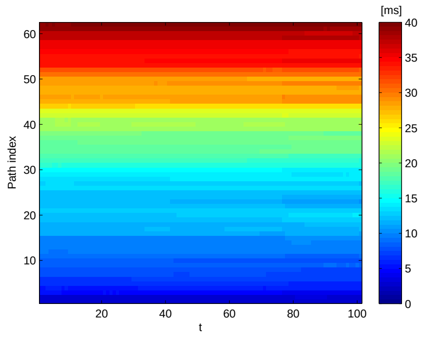
(a) True map



(b) Kriging



(c) Wavelets



(d) KKF

Figure 6.2: True and predicted delay map for 62 paths in the Internet2 network over in interval of 100 minutes.

Predictions are performed using measurements over an interval of 100 minutes on 10 random paths (same paths are used throughout the considered interval), and the delays are predicted on the remaining 62 paths are reported. In these maps, paths are arranged in increasing order according to the true delay at time $t = 1$. It can be seen that the map produced by the kriging and compressive sensing approaches are very different from the true map. In contrast, the map obtained when using the KKF is close to the true map. In particular, observe that the delays of several paths change slightly around $t = 80$ in Figure 6.2(a). However, of the three maps, this change is only discernible in the KKF map in 6.2(d). The delay predictions provided by the KKF are thus sufficiently accurate for human inspection at control centers, even when monitoring a few paths.

For a more detailed analysis of the different delay prediction approaches, the normalized

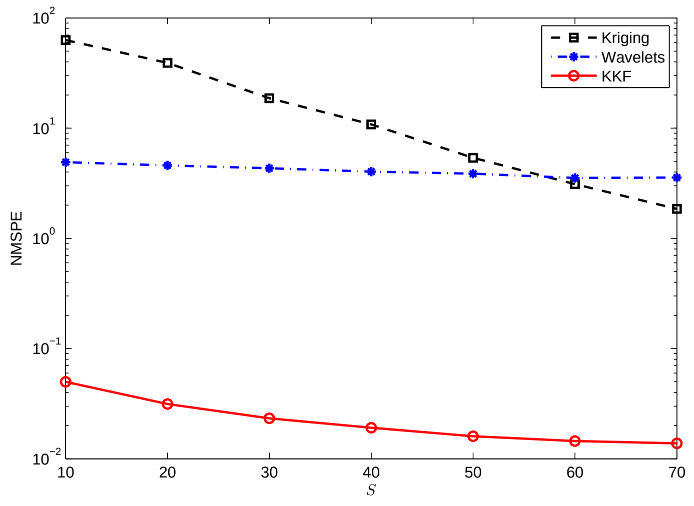


Figure 6.3: NMSPE as a function of S, Internet2 network with random path selection.

mean-square prediction error (NMSPE) is considered. It is defined as

$$\text{NMSPE} := \frac{1}{(t_P - t_L)(P-S)} \sum_{t=t_L+1}^{t_P} \|\hat{\mathbf{y}}_{\bar{s}}(t) - \mathbf{y}_{\bar{s}}(t)\|_2^2 . \tag{6.23}$$

The prediction performance of the three algorithms is first assessed by using delay measurements on randomly selected paths for each t. The (same) randomly selected paths are used for all three approaches. Figure 6.3 depicts the NMSPE as a function of S, the number of paths on which delays are measured. Clearly, the KKF markedly outperforms the other two approaches across the entire range of S. As expected [30], the compressive sampling-based approach provides a more accurate prediction than network kriging.

Next, the performance of the three algorithms is analyzed for the case when paths for delay measurement are selected optimally. For the network kriging and the wavelet-based approaches, the optimal paths are obtained according to the selection procedures provided in [29] and [30], respectively. As pointed out in [30], performance of the wavelet-based approach can be improved by capitalizing on temporal correlations. This is done by solving (6.22) using measurements from $\tau = 5$ consecutive time slots in a batch form. The temporal correlation is set to 0.5 and the optimal paths are obtained again using the selection strategy outlined in [30]. For the KKF, optimal paths are selected in an online fashion using Algorithm 1. Again, a sig-

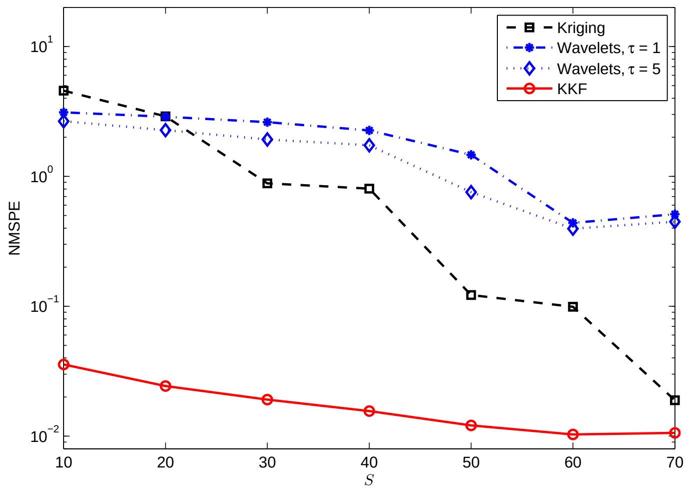


Figure 6.4: NMSPE as a function of S, Internet2 network with optimal path selection.

nificantly more accurate prediction of the path delays for the entire range of S is obtained via the KKF.

6.4.2 NZ-AMP Delay Data

The KKF algorithm is tested here using delay data from NZ-AMP. The project continuously runs `ICMP` and scamper to determine the topology and delays between a set of nodes in New Zealand. The data collected for this chapter consist of end-to-end delays measured every ten minutes over the month of August 2011. The network has a total of 186 paths, whose delays range from almost constant to highly variable, at times reaching up to 250ms.

In Figure 6.5, the NMSPE as a function of S is reported, for the case where paths that are to be measured are chosen randomly. Again, same paths are used for the three considered schemes. The KKF provides a markedly lower prediction error also for the NZ-AMP delay data. On the other hand, Figure 6.6 shows the NMSPE on optimally selected paths for all three schemes. The KKF performs relatively better than the competing schemes for this data set as well. Observe though that the actual values of the NMSPE incurred for this dataset is at least an order of magnitude higher than those in the Internet2 dataset. Indeed, given the high variability in the data, it is possible to improve upon the prediction accuracy of KKF

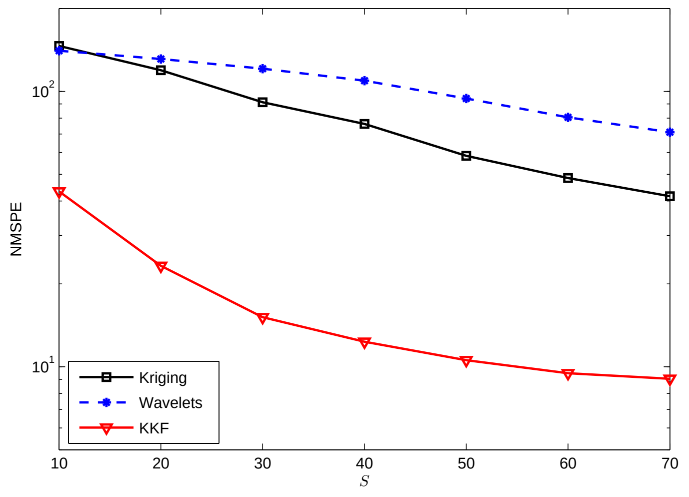


Figure 6.5: NMSPE as a function of S, NZ-AMP network with random path selection.

by training it better. This is showcased by the considerably lower prediction error curve for training interval t_L=2,000 shown in Figure 6.6.

While the NMSPE is useful for characterizing the average performance, network operators are also interested in the prediction accuracy over the entire range of delay values. Towards this end, Figure 6.7 shows the scatter plots of $\hat{\mathbf{y}}_{\bar{s}}(t)$ versus $\mathbf{y}_{\bar{s}}(t)$ for all t and $S = 30$ optimally selected paths. The points cluster around the 45-degree line $\hat{\mathbf{y}}_{\bar{s}}(t) = \mathbf{y}_{\bar{s}}(t)$, and the thinner the "cloud" of points is, the more accurate the estimates are. Indeed, it can be seen that the points generated from the KKF estimates are crammed in a very close area around the 45-degree line, and accurate estimates are produced for the entire range of experienced delays. Furthermore, the scatter plots corroborate the unbiasedness of the KKF predictor.

6.5 Conclusion

The present chapter develops a spatio-temporal prediction approach to track and predict network-wide path delays using measurements on only a few paths. The proposed algorithm adapts a kriged Kalman filter that exploits both topological as well as historical data. The framework also allows for the use of submodular optimization in the selection of optimal delay measure-

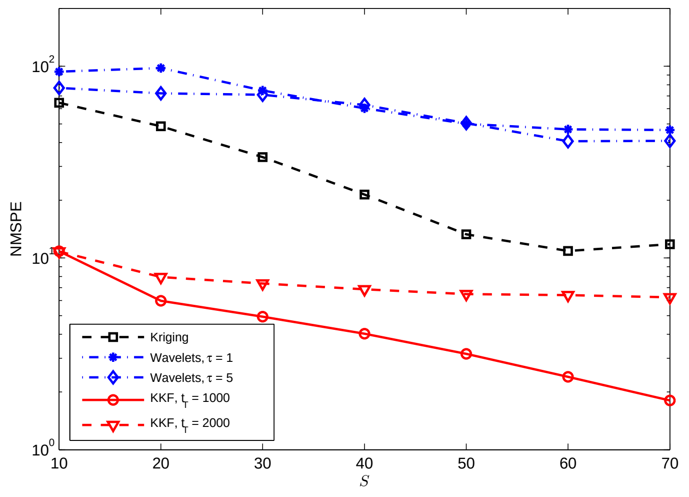


Figure 6.6: NMSPE as a function of S, NZ-AMP network with optimal path selection.

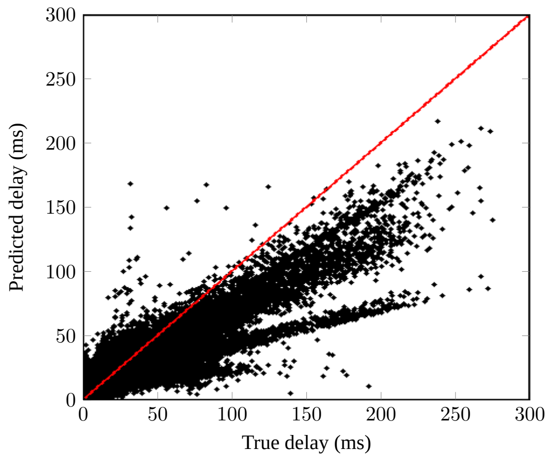
ment locations. The problem of path selection is formulated for different types of constraints on the set of selected paths, and solved in an online fashion to near-optimality. The resulting predictor is validated on two datasets with different delay profiles, and is shown to substantially outperform competing alternatives.

6.A Error Covariance Matrix

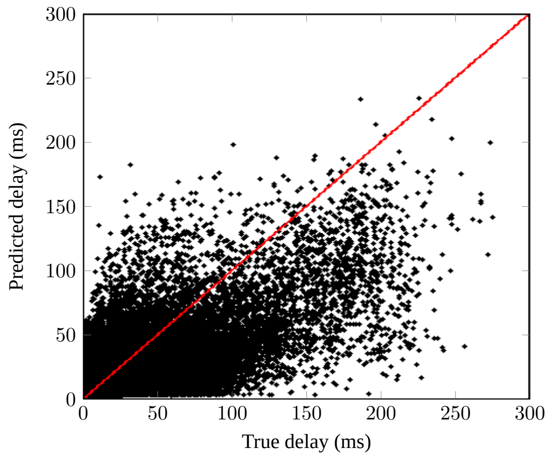
Towards deriving an expression for $\mathbf{M}_{\bar{s}}^{\mathbf{y}}(t)$, observe that the prediction error can be written as

$$\begin{aligned}\mathbf{y}_{\bar{s}}(t) - \hat{\mathbf{y}}_{\bar{s}}(t) &= \bar{\mathbf{S}}(t)\boldsymbol{\chi}(t) + \bar{\mathbf{S}}(t)\boldsymbol{\nu}(t) + \boldsymbol{\epsilon}_{\bar{s}}(t) - \bar{\mathbf{S}}(t)\hat{\boldsymbol{\chi}}(t) \\ &\quad - \bar{\mathbf{S}}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)\left(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right)^{-1}[\mathbf{y}_s(t) - \mathbf{S}(t)\hat{\boldsymbol{\chi}}(t)] \qquad (6.24) \\ &= \bar{\mathbf{S}}(t)(\boldsymbol{\chi}(t) - \hat{\boldsymbol{\chi}}(t) + \boldsymbol{\nu}(t)) + \boldsymbol{\epsilon}_{\bar{s}}(t) \\ &\quad - \bar{\mathbf{S}}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)\left(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right)^{-1}[\mathbf{S}(t)(\boldsymbol{\chi}(t) - \hat{\boldsymbol{\chi}}(t) + \boldsymbol{\nu}(t)) + \boldsymbol{\epsilon}_s(t)]\,. \end{aligned}$$

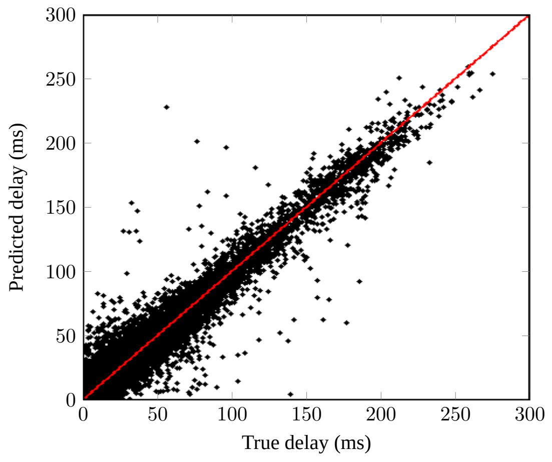
(6.25)



(a) Kriging



(b) Wavelets



(c) KKF

Figure 6.7: Scatter plot for the NZ-AMP network, $S = 30$ with optimal path selection.

Using (6.4a), the term $\boldsymbol{\chi}(t) - \hat{\boldsymbol{\chi}}(t)$ can be written as

$$\begin{aligned}\boldsymbol{\chi}(t) - \hat{\boldsymbol{\chi}}(t) &= \boldsymbol{\chi}(t) - \hat{\boldsymbol{\chi}}(t-1) - \mathbf{K}(t)\left[\mathbf{S}(t)(\boldsymbol{\chi}(t) + \boldsymbol{\nu}(t)) + \boldsymbol{\epsilon}_s(t) - \mathbf{S}(t)\hat{\boldsymbol{\chi}}(t-1)\right] \\ &= \boldsymbol{\chi}(t) - \hat{\boldsymbol{\chi}}(t-1) + \mathbf{K}(t)\mathbf{S}(t)(\boldsymbol{\chi}(t) - \hat{\boldsymbol{\chi}}(t-1) + \boldsymbol{\nu}(t)) + \mathbf{K}(t)\boldsymbol{\epsilon}_s(t) \\ &= (\mathbf{I}_P - \mathbf{K}(t)\mathbf{S}(t))\tilde{\boldsymbol{\chi}}(t) - \mathbf{K}(t)\mathbf{S}(t)\boldsymbol{\nu}(t) - \mathbf{K}(t)\boldsymbol{\epsilon}_s(t) \qquad (6.26)\end{aligned}$$

where $\tilde{\boldsymbol{\chi}}(t) := \boldsymbol{\chi}(t) - \hat{\boldsymbol{\chi}}(t-1)$. Substituting (6.26) in (6.25), it follows that

$$\begin{aligned}\mathbf{y}_{\bar{s}}(t) - \hat{\mathbf{y}}_{\bar{s}}(t) = {}& \bar{\mathbf{S}}(t)(\mathbf{I}_P - \mathbf{K}(t)\mathbf{S}(t))(\tilde{\boldsymbol{\chi}}(t) + \boldsymbol{\nu}(t)) - \bar{\mathbf{S}}(t)\mathbf{K}(t)\boldsymbol{\epsilon}_s(t) + \boldsymbol{\epsilon}_{\bar{s}}(t) \\ & - \bar{\mathbf{S}}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)\left(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right)^{-1} \\ & \quad \times \left[\mathbf{S}(t)(\mathbf{I}_P - \mathbf{K}(t)\mathbf{S}(t))(\tilde{\boldsymbol{\chi}}(t) + \boldsymbol{\nu}(t)) - \mathbf{S}(t)\mathbf{K}(t)\boldsymbol{\epsilon}_s(t) + \boldsymbol{\epsilon}_s(t)\right] \\ & \qquad (6.27)\end{aligned}$$

$$\begin{aligned}= {}& \bar{\mathbf{S}}(t)(\mathbf{I}_P - \mathbf{K}(t)\mathbf{S}(t))(\tilde{\boldsymbol{\chi}}(t) + \boldsymbol{\nu}(t)) \\ & - \bar{\mathbf{S}}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)\left(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right)^{-1}\mathbf{S}(t)(\mathbf{I}_P - \mathbf{K}(t)\mathbf{S}(t))(\tilde{\boldsymbol{\chi}}(t) + \boldsymbol{\nu}(t)) \\ & - \bar{\mathbf{S}}(t)\mathbf{K}(t)\boldsymbol{\epsilon}_s(t) - \bar{\mathbf{S}}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)\left(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right)^{-1}(\mathbf{I}_S - \mathbf{S}(t)\mathbf{K}(t))\boldsymbol{\epsilon}_s(t) \\ & + \boldsymbol{\epsilon}_{\bar{s}}(t) \qquad (6.28)\end{aligned}$$

which, after some manipulations, can be expressed as

$$\mathbf{y}_{\bar{s}}(t) - \hat{\mathbf{y}}_{\bar{s}}(t) = \bar{\mathbf{S}}(t)(\mathbf{I}_P - \mathbf{Q}(t)\mathbf{S}(t))(\tilde{\boldsymbol{\chi}}(t) + \boldsymbol{\nu}(t)) + \mathbf{Q}(t)\boldsymbol{\epsilon}_s(t) + \boldsymbol{\epsilon}_{\bar{s}}(t) \qquad (6.29)$$

where

$$\begin{aligned}\mathbf{Q}(t) := {}& \mathbf{K}(t) + \mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}(t)(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S)^{-1} \\ & - \mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}(t)(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S)^{-1}\mathbf{S}(t)\mathbf{K}(t). \qquad (6.30)\end{aligned}$$

Next, substituting for $\mathbf{K}(t)$ from (6.5), the expression for $\mathbf{Q}(t)$ simplifies to

$$\begin{aligned}\mathbf{Q}(t) = {}& (\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}})\mathbf{S}^T(t)\left[\mathbf{S}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}})\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right]^{-1} \\ & + \mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S)^{-1} \\ & - \mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t)(\mathbf{S}(t)\mathbf{C}_{\boldsymbol{\nu}}\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S)^{-1}\mathbf{S}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}})\mathbf{S}^T(t) \\ & \quad \times \left[\mathbf{S}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}})\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right]^{-1} \qquad (6.31) \\ = {}& (\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}})\mathbf{S}^T(t)\left[\mathbf{S}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}})\mathbf{S}^T(t) + \sigma^2\mathbf{I}_S\right]^{-1} \qquad (6.32)\end{aligned}$$

Utilizing the fact that $\tilde{\boldsymbol{\chi}}(t)$, $\boldsymbol{\nu}(t)$, $\boldsymbol{\epsilon}_s(t)$, and $\boldsymbol{\epsilon}_{\bar{s}}(t)$ are mutually uncorrelated, with $\mathbb{E}\left[\tilde{\boldsymbol{\chi}}(t)\tilde{\boldsymbol{\chi}}^T(t)\right] := \mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}}$, the error covariance matrix $\mathbf{M}_{\bar{s}}^{\mathbf{y}}(t)$ becomes

$$\begin{aligned}
\mathbf{M}_{\bar{s}}^{\mathbf{y}}(t) &= \mathbb{E}\left[(\mathbf{y}_{\bar{s}}(t) - \hat{\mathbf{y}}_{\bar{s}}(t))(\mathbf{y}_{\bar{s}}(t) - \hat{\mathbf{y}}_{\bar{s}}(t))^T\right] && (6.33)\\
&= \bar{\mathbf{S}}(t)(\mathbf{I}_P - \mathbf{Q}(t)\mathbf{S}(t))(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\nu}} + \mathbf{C}_{\boldsymbol{\eta}})(\mathbf{I}_P - \mathbf{S}^T(t)\mathbf{Q}^T(t))\bar{\mathbf{S}}^T(t)\\
&\quad + \sigma^2\bar{\mathbf{S}}(t)\mathbf{Q}(t)\mathbf{Q}^T(t)\bar{\mathbf{S}}^T(t) + \sigma^2\mathbf{I}_{P-S} && (6.34)\\
&= \bar{\mathbf{S}}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\nu}} + \mathbf{C}_{\boldsymbol{\eta}})\bar{\mathbf{S}}^T(t) - 2\bar{\mathbf{S}}(t)\mathbf{Q}(t)\mathbf{S}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\nu}} + \mathbf{C}_{\boldsymbol{\eta}})\bar{\mathbf{S}}^T(t)\\
&\quad + \bar{\mathbf{S}}(t)\mathbf{Q}(t)\mathbf{S}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}})\mathbf{S}^T(t)\mathbf{Q}^T(t)\bar{\mathbf{S}}^T(t) + \sigma^2\bar{\mathbf{S}}(t)\mathbf{Q}(t)\mathbf{Q}^T(t)\bar{\mathbf{S}}^T(t)\\
&\quad + \sigma^2\mathbf{I}_{P-S} && (6.35)\\
&= \bar{\mathbf{S}}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\nu}} + \mathbf{C}_{\boldsymbol{\eta}})\bar{\mathbf{S}}^T(t) - \bar{\mathbf{S}}(t)\mathbf{Q}(t)\mathbf{S}(t)(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\nu}} + \mathbf{C}_{\boldsymbol{\eta}})\bar{\mathbf{S}}^T(t)\\
&\quad + \sigma^2\mathbf{I}_{P-S}. && (6.36)
\end{aligned}$$

Substituting for $\mathbf{Q}(t)$ [cf. (6.32)] in (6.36), and using the Woodbury matrix identity [67], the final expression for $\mathbf{M}_{\bar{s}}^{\mathbf{y}}(t)$ becomes

$$\mathbf{M}_{\bar{s}}^{\mathbf{y}}(t) = \sigma^2\mathbf{I}_{P-S} + \bar{\mathbf{S}}(t)\left[\left(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\nu}} + \mathbf{C}_{\boldsymbol{\eta}}\right)^{-1} + \frac{1}{\sigma^2}\mathbf{S}^T(t)\mathbf{S}(t)\right]^{-1}\bar{\mathbf{S}}^T(t)\,. \quad (6.37)$$

6.B Proof of Monotonicity and Supermodularity of f

Let $\boldsymbol{\Phi} := \frac{1}{\sigma^2}(\mathbf{M}(t-1) + \mathbf{C}_{\boldsymbol{\eta}} + \mathbf{C}_{\boldsymbol{\nu}})$, and observe that f can be written as

$$\begin{aligned}
f(\mathcal{S}) &= \log(\sigma^2) + \log\det\left[\mathbf{I}_{P-S} + \bar{\mathbf{S}}(\boldsymbol{\Phi}^{-1} + \mathbf{S}^T\mathbf{S})^{-1}\bar{\mathbf{S}}^T\right] && (6.38a)\\
&= \log(\sigma^2) + \log\det\left[\mathbf{I}_P + \bar{\mathbf{S}}^T\bar{\mathbf{S}}(\boldsymbol{\Phi}^{-1} + \mathbf{S}^T\mathbf{S})^{-1}\right] && (6.38b)\\
&= \log(\sigma^2) + \log\det\left[\boldsymbol{\Phi}^{-1} + \mathbf{S}^T\mathbf{S} + \bar{\mathbf{S}}^T\bar{\mathbf{S}}\right] + \log\det\left[(\boldsymbol{\Phi}^{-1} + \mathbf{S}^T\mathbf{S})^{-1}\right] && (6.38c)
\end{aligned}$$

where (6.38b) follows from Sylvester's theorem for determinants [67].

Observing that $\bar{\mathbf{S}}^T\bar{\mathbf{S}} + \mathbf{S}^T\mathbf{S} = \mathbf{I}_P$, it is possible to write $f(\mathcal{S})$ as

$$f(\mathcal{S}) = \log(\sigma^2) + \log\det(\boldsymbol{\Phi}^{-1} + \mathbf{I}_P) - \log\det\left(\boldsymbol{\Phi}^{-1} + \mathbf{S}^T\mathbf{S}\right). \quad (6.39)$$

Next, consider the decomposition $\mathbf{\Phi} = \mathbf{U}\mathbf{U}^T$, and define the shifted function

$$\begin{aligned} h(\mathcal{S}) &:= f(\mathcal{S}) - \log(\sigma^2) - \log\det\left(\mathbf{\Phi} + \mathbf{I}_P\right) && (6.40a) \\ &= -\log\det(\mathbf{I}_P + \mathbf{S}^T\mathbf{S}\mathbf{\Phi}) && (6.40b) \\ &= -\log\det\left[\mathbf{I}_S + (\mathbf{S}\mathbf{U})(\mathbf{S}\mathbf{U})^T\right] && (6.40c) \end{aligned}$$

where Sylvester's theorem has again been used in (6.40c). Finally, it is well known that a function of the form $\log\det(\mathbf{I}_P + (\mathbf{S}\mathbf{U})^T(\mathbf{S}\mathbf{U}))$ is non-decreasing and submodular (see e.g., [5]), which allows one to deduce that $f(\mathcal{S})$ is non-increasing and supermodular. Note further that the greedy approach from [114] can be used on $h(\mathcal{S})$ by defining $h(\emptyset) = 0$.

Chapter 7

Summary and Future Work

This thesis touched upon several key monitoring and resource allocation problems present in communication networks. Chapters 2–5 leveraged the idea of network coding to design wireless network protocols for information collection and dissemination in resource-constrained ad hoc networks. Towards achieving this goal, a cross-layer design approach was pursued, and network codes were optimized jointly with protocols operating at application, medium access control (MAC), and physical (PHY) layers.

Chapter 2 considered wireless fading networks, where network coding can be optimally integrated into the protocol stack using a dual decomposition method. Leveraging this result, an adaptation of the subgradient method suitable for network control was also developed. The method is asynchronous, because the physical layer is allowed to return its contribution to the subgradient vector with some delay.

In Chapter 3, network coding was introduced for use with Aloha-based MAC and PHY layers, which are attractive in their simplicity. Although the overall optimization problem is still non-convex, successive approximation is adopted to realize efficient network coding algorithms. The idea was also extended to create a separable structure in the problem, enabling the dual decomposition technique to yield a distributed solution. The algorithm is thus applicable for large networks, and amenable to online implementation.

Benefits of network coding also extend to *QoS-constrained* scenarios, such as in real-time and streaming media applications. Modeling constraints on packet deadlines is the key challenge here, and Chapter 4 puts forth constant-factor approximations to this end. The setup was also analyzed from an integer programming perspective, and a set of valid inequalities was developed and used to obtain a linear programming based upper bound on the throughput.

Chapter 5 dealt with sensor networks where the observed data is correlated across nodes, and network coding can both compress and communicate the data to a collection agent. An efficient decoding scheme for this network-compressive coding scheme was developed, yielding network-wide energy savings and increase in the network lifetime. Error exponents and simulation results were provided to delineate, quantify, and test the interplay between the estimation error, tolerable distortion, alphabet size, and communication cost.

The second part of this thesis advocated dynamic network cartography as tool for monitoring and prediction of the evolving network state. Chapter 6 developed a spatio-temporal prediction approach to track and predict network-wide path delays using measurements on only a few paths. The proposed framework not only exploits both topological and historical data, but also allows for the use of submodular optimization in the selection of optimal delay measurement locations.

Before concluding this thesis, the remainder of this chapter describes future research directions which build on the framework and tools developed hitherto.

7.1 Dictionary Learning for Traffic Maps with Missing Data

An interesting extension of the network cartography framework involves inference and prediction of traffic volumes on links (also referred to as link counts) in IP networks. Link counts are one of the primary indicators of instantaneous network health, and serve as the basic ingredient for more complex management tasks such as intrusion detection, capacity provisioning, and network planning. Information about link utilization is typically available to network operators through off-the-shelf tools such as the SNMP. Missing entries in the link counts may however skew the network operator's perspective. Packets may be dropped in SNMP for instance, if some links become congested, rendering link count information for those links more important, as well as less available [99, 155].

Let the $L \times 1$ vector $\mathbf{y}(t)$ collect the link counts on L network links at a given time t.

Typically, only an $M \times 1$ sub-vector $\mathbf{y}^o(t)$ with $M < L$ entries is observed at any time, and the goal is to predict the unobserved $(L - M) \times 1$ sub-vector $\mathbf{y}^u(t)$ using historical data and topological information. In principle, missing link loads could be estimated if the matrix of flow volumes between all origin-destination (OD) pairs $\mathbf{X}(t)$ were known. Upon defining the $F \times 1$ vector $\mathbf{x}(t) := \mathrm{vec}(\mathbf{X}(t))$ as the vectorized traffic matrix, it readily follows that

$$\mathbf{y}^u(t) = \mathbf{R}^u\mathbf{x}(t)\,, \quad t = 1, \ldots, T \tag{7.1}$$

where the routing matrix entry $[\mathbf{R}^u]_{\ell,f}$ equals one if flow f passes through link ℓ, and zero otherwise, and $L \ll F$. However, measuring flow volumes is even more difficult, and in practice, $\mathbf{x}(t)$ is itself estimated from $\mathbf{y}^o(t)$ and $\mathbf{R}^o$. Since traffic matrix estimation is an *underdetermined* problem, most proposed approaches use specific priors or regularization techniques, and tacitly rely on the stationarity of $\mathbf{x}(t)$; see e.g., [155, 176, 177] and references therein.

As future work, a more direct approach to predicting link counts is feasible by postulating the over-complete representation $\mathbf{y} = \mathcal{B}\,\mathbf{s}$ over a basis matrix $\mathcal{B}$, with columns $\{\mathbf{b}_p\}_{p=1}^P$ constrained to have unit norm (which avoids scaling ambiguity), and a sparse coefficient vector $\mathbf{s}$. Given $\mathcal{B}$ and the $M \times 1$ vector of observed link counts $\mathbf{y}^o$, contemporary compressive sampling tools [6, 69, 157, 158] can be adopted to estimate the missing $L - M$ link counts $\mathbf{y}^u$. Consider partitioning the basis matrix as $\mathcal{B} := [(\mathcal{B}^o)'\ (\mathcal{B}^u)']'$, where $\mathcal{B}^o$ corresponds to rows of measured link counts, $\mathcal{B}^u$ to rows of missing ones, and $(\cdot)'$ stands for transposition. During the *operational phase*, the sparse representation for $\mathbf{y} := [(\mathbf{y}^o)'\ (\mathbf{y}^u)']'$ can be estimated using the least-absolute shrinkage and selection operator (Lasso) [157], as

$$\hat{\mathbf{s}} := \arg\min_{\mathbf{s}} \|\mathbf{y}^o - \mathcal{B}^o\mathbf{s}\|_2^2 + \lambda\|\mathbf{s}\|_1 \tag{7.2}$$

where the tuning parameter $\lambda > 0$ controls the sparsity of $\hat{\mathbf{s}}$, and can be chosen using standard cross-validation techniques [68]. Once $\hat{\mathbf{s}}$ is available, the missing link counts are predicted as $\hat{\mathbf{y}}^u = \mathcal{B}^u\hat{\mathbf{s}}$. It is evident that for a given sparsity level dictated by $\lambda\|\mathbf{s}\|_1$, the quality of the predicted $\hat{\mathbf{y}}^u$ depends on $\mathcal{B}$. Bases comprising columns $\{\mathbf{b}_i\}$ that explain well the link counts across the network will lead to improved predictions of $\mathbf{y}^u$. Thus, the selection of $\mathcal{B}$ must be data-driven thereby shaping the columns $\{\mathbf{b}_p\}_{p=1}^P$ to the link-count prediction task at hand.

Choosing an over-complete basis on which a signal admits a sparse representation has led to exciting advancements in the area of *dictionary learning* [102, 103, 116, 158]. In its canonical form, dictionary learning seeks a basis $\mathcal{B}$ so that training data $\mathcal{T} := \{\mathbf{y}(t)\}_{t=1}^{T}$ is well approximated as $\mathbf{y}(t) \approx \mathcal{B}\mathbf{s}(t)$, $\forall t = 1, \ldots, T$, where $\mathbf{s}(t)$ is a *sparse* coefficient vector. Given historical link counts $\mathcal{T}$, it is possible to apply results from dictionary learning to construct $\mathcal{B}$. However, this would require T measurements of the link counts at all network links. Gathering $\mathcal{T}$ quickly becomes infeasible as the network size grows, thereby rendering canonical dictionary learning impractical. To circumvent this challenge, the idea is to capitalize on semi-supervised learning and manifold regularization [9, 125].

Consider the historical data set $\mathcal{T}_M := \{\mathbf{y}^o(t)\}_{t=1}^{T}$ formed by observed link counts at $M < L$ links. Each $M \times 1$ measurement vector is $\mathbf{y}_t^o := \mathbf{J}_t\mathbf{y}(t)$, where $\mathbf{J}_t$ is an $M \times L$ binary matrix choosing the M measured link counts for the t-th measurement. To enable learning $\mathcal{B}$ from $\mathcal{T}_M$ instead of $\mathcal{T}$, it is assumed that the $L \times Q$ network routing matrix $\mathbf{R}$ is available, where Q denotes the number of OD pairs in the network. Each column of $\mathbf{R}$ contains the routing path for a given OD pair of nodes. Using $\mathbf{R}$, it is possible to construct an auxiliary weighted graph $\mathcal{G}$ with L nodes, one corresponding to each network link. The edge weights for all links in the graph are subsumed by the off-diagonal entries of the Gram matrix $\mathbf{G} := \mathbf{R}\mathbf{R}'$. The data-driven basis $\mathcal{B}$ is obtained during a *training phase* as

$$(\{\hat{\mathbf{s}}(t)\}_{t=1}^{T}, \mathcal{B}) = \underset{\substack{\{\mathbf{s}_t\}_{t=1}^{T}, \\ \mathbf{B}:\|\mathbf{b}_p\|_2 \leq 1}}{\arg\min} \sum_{t=1}^{T} \|\mathbf{y}_t^o - \mathbf{J}_t\mathbf{B}\mathbf{s}_t\|_2^2 + \lambda_s \sum_{t=1}^{T} \|\mathbf{s}_t\|_1 + \lambda_g \sum_{t=1}^{T} \mathbf{s}_t'\mathbf{B}'\mathbf{L}\mathbf{B}\mathbf{s}_t \quad (7.3)$$

where $\mathbf{L}$ is the Laplacian matrix of $\mathcal{G}$, and $\lambda_s, \lambda_g > 0$ are tuning parameters. The regularization terms in (7.3) control both sparsity in the expansion coefficients through the ℓ_1-norm, but also smoothness of the link-count predictions via $\mathbf{L}$. The optimization problem is nonconvex; however, with $\mathbf{B}$ fixed, the problem is convex w.r.t. $\{\mathbf{s}_t\}$, and vice versa, allowing one to employ coordinate descent solvers. Future research directions include convergence analysis of the coordinate descent algorithm, development of online prediction and learning approaches along the lines of recursive least-squares, and incorporation of temporal correlations in the formulation (7.3).

7.2 Joint Rate and Power Control for Coded CR Networks

Cognitive radio (CR) has recently been recognized as an emerging disruptive technology that holds great potential to enhance spectrum utilization [70, 120]. A major component of the CR technology has been dynamic spectrum access, in which network users opportunistically gain wireless access to licensed frequency bands without causing harmful interference to incumbent primary users (PUs) [52]. When designing CR networks however, pertinent approaches can not rely on the accumulated knowledge in conventional ad hoc networks, but rather have to account for the peculiarities of hierarchical access schemes [179], and autonomous interference management.

This motivates the development of resource allocation schemes that rely on the sensing result [165] and facilitate the inclusion of capacity achieving protocols such as network coding [97]. A fruitful direction to this end is a cross-layer design framework in order to jointly optimize power and rate allocations in coded CR networks in the presence of channel uncertainty induced by both shadowing and small-scale fading. Channel uncertainty-aware CR-specific constraints can be incorporated directly into the optimization formulation, so as to yield a resource allocation algorithm rooted at the PHY layer. Of particular interest is delay-limited CR traffic, where QoS requirements are severe, and channel outages are common. The idea is to formulate a joint coding/routing rate and power allocation problem, which maximizes the network-wide utility while constraining the outage probabilities and average interference to the PU networks.

The primary challenge encountered when designing systems with outage probability constraints is that the resulting problem formulations are too complex for realistic channel distributions [54]. Related work in this context includes [81], where only small-scale fading is considered. Future work can also include examination of these outage probability distributions with the aim of obtaining an approximate convex problem, amenable to efficient solution. The research issue here is to properly utilize the result that allows approximating a sum of log-normal random variables with a single log-normal random variable; see e.g., [41, 53, 54, 108]. Such a scheme should then be able to adapt the network and PHY layer parameters to the propagation environment.

Bibliography

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

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[3] M. H. Amerimehr, B. H. Khalaj, and P. M. Crespo, “A distributed cross-layer optimization method for multicast in interference-limited multihop wireless networks,” *EURASIP Journal on Wireless Communication and Networking*, vol. 2008, June 2008.

[4] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, ser. Prentice-Hall Information and System Sciences Series. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[5] F. Bach, “Learning with submodular functions: A convex optimization perspective,” *Foundations and Trends in Machine Learning*, 2012. [Online]. Available: http://arxiv.org/abs/1111.6453

[6] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, “Compressive wireless sensing,” in *Proc. International Conference on Information Processing in Sensor Networks*, ser. IPSN ’06, 2006, pp. 134–142.

[7] D. Baron, S. Sarvotham, and R. Baraniuk, “Bayesian compressive sensing via belief propagation,” *IEEE Transactions on Signal Processing*, vol. 58, no. 1, pp. 269–280, Jan. 2010.

[8] H. Bartz, T. Lutz, C. Hausl, and J. Barros, “Practical network coding with resilient subspace codes,” in *Proc. of the 19th International Conference on Communications and Networks*, Zurich, Switzerland, Aug. 2010.

[9] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,” *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, Dec. 2006.

[10] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, MA: Athena Scientific, 1998.

[11] ——, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[12] D. P. Bertsekas, A. Nedić, and A. Ozdaglar, *Convex Analysis and Optimization*. Belmont, MA: Athena Scientific, 2003.

[13] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.

[14] K. Bharath-Kumar and J. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Transactions on Communications*, vol. COM-31, no. 3, pp. 343–351, Mar. 1983.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[16] D. Blackwell, "On a theorem of Lyapunov," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 112–114, Mar. 1951.

[17] C. J. Bovy, H. T. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. van Mieghem, "Analysis of end-to-end delay measurements in Internet," in *Proc. of the Passive and Active Measurement Workshop*, Fort Collins, CO, Apr. 2002.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY: Cambridge University Press, 2004.

[19] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proc. of the IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2936–2940.

[20] R. E. Burkard, K. Dlaska, and B. Klinz, "The quickest flow problem," *Mathematical Methods of Operations Research*, vol. 37, no. 1, pp. 31–58, Feb. 1993.

[21] N. Cai and R. W. Yeung, "Secure network coding on a wiretap network," *IEEE Transactions on Information Theory*, vol. 57, no. 1, pp. 424–435, Jan. 2011.

[22] M. Cetin, L. Chen, J. W. Fisher III, A. T. Ihler, R. L. Moses, M. J. Wainwright, and A. S. Willsky, "Distributed fusion in sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 42–55, July 2006.

[23] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multicast with network coding," in *Proc. of the IEEE INFOCOM*, Anchorage, AK, May 2007, pp. 1163–1171.

[24] Y. Chen, D. Bindel, and R. H. Katz, "Tomography-based overlay network monitoring," in *Proc. of the ACM SIGCOMM Internet Measurement Conference*. ACM press, Oct. 2003.

[25] M. Chiang, "Geometric programming for communication systems," *Communications and Information Theory*, vol. 2, no. 1/2, pp. 1–154, July 2005.

[26] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[27] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. of the 41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Oct. 2003, pp. 40–49.

[28] P. Chou and Y. Wu, "Network coding for the Internet and wireless networks," *IEEE Signal Processing Magazine*, vol. 24, no. 5, pp. 77–85, Sep. 2007.

[29] D. B. Chua, E. D. Kolaczyk, and M. Crovella, "Network kriging," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2263–2272, Dec. 2006.

[30] M. Coates, Y. Pointurier, and M. Rabbat, "Compressed network monitoring for IP and all-optical networks," in *Proc. of the ACM Internet Measurement Conference*, San Diego, CA, Oct. 2007.

[31] R. R. Coifman and M. Maggioni, "Diffusion wavelets," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 53–94, 2006.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill, Cambridge, MA, 2001.

[33] J. Cortés, "Distributed kriged Kalman filter for spatial estimation," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2816–2827, Dec. 2009.

[34] R. Costa, D. Munaretto, J. Widmer, and J. Barros, "Informed network coding for minimum decoding delay," in *Proc. of the International Conference on Mobile Ad hoc and Sensor Networks*, Atlanta, GA, Sep. 2008, pp. 80–91.

[35] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience, 2006.

[36] N. Cressie, "The origins of kriging," *Mathematical Geology*, vol. 22, no. 3, pp. 239–252, 1990.

[37] I. Csiszar, “The method of types,” *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2505–2523, Oct. 1998.

[38] T. Cui, L. Chen, and T. Ho, “Energy efficient opportunistic network coding for wireless networks,” in *Proc. of the IEEE INFOCOM*, Phoenix, AZ, Apr. 2008, pp. 1022–1030.

[39] ——, “On distributed scheduling in wireless networks exploiting broadcast and network coding,” *IEEE Transactions on Communications*, vol. 58, no. 4, pp. 1223–1234, Apr. 2010.

[40] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, “Vivaldi: A decentralized network coordinate system,” in *Proc. of the ACM SIGCOMM*, Portland, Oregon, USA, 2004, pp. 15–26.

[41] E. Dall’Anese, S.-J. Kim, G. B. Giannakis, and S. Pupolin, “Power control for cognitive radio networks under channel uncertainty,” *IEEE Transactions on Wireless Communications*, vol. 10, no. 10, pp. 3541–3551, Dec. 2011.

[42] E. Dall’Anese, S.-J. Kim, and G. Giannakis, “Channel gain map tracking via distributed kriging,” *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 1205–1211, Mar. 2011.

[43] A. Das and D. Kempe, “Algorithms for subset selection in linear regression,” in *Proc. of the ACM Symposium on Theory of Computing*, Victoria, British Columbia, Canada, May 2008, pp. 45–54.

[44] A. G. Dimakis, P. B. Godfrey, M. Wainwright, and K. Ramchandran, “Network coding for distributed storage,” in *Proc. of the IEEE INFOCOM*, Anchorage, AK, Mar. 2007, pp. 2000–2008.

[45] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[46] N. Dinculeanu, *Vector Measures*. Oxford, U.K.: Pergamon Press, 1967.

[47] S. C. Draper and S. Malekpour, “Compressed sensing over finite fields,” in *Proc. of the International Symposium Information Theory*, Seoul, Korea, July 2009, pp. 669–673.

[48] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge University Press, 2011.

[49] E. Erez and M. Feder, “Convolutional network codes,” in *Proc. of the International Symposium on Information Theory*, June 2004, p. 146.

[50] E. Erez, M. Effros, and T. Ho, “Network codes with deadlines,” in *Proc. of the 46th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sep. 2008, pp. 339–346.

[51] A. Eryilmaz, A. Ozdaglar, M. Médard, and E. Ahmed, “On the delay and throughput gains of coding in unreliable networks,” *IEEE Transactions on Information Theory*, vol. 54, no. 12, pp. 5511–5524, 2008.

[52] “Fcc spectrum policy task force report,” ET-Docket 02-135, FCC, Nov. 2002.

[53] L. F. Fenton, “The sum of lognormal probability distributions in scatter transmission systems,” *IRE Trans. Communication Syst.*, vol. 8, no. 1, pp. 57–67, Mar. 1960.

[54] C. Fischione, M. D’Angelo, and M. Butussi, “Utility maximization via power and rate allocation with outage constraints in nakagami-lognormal channels,” *IEEE Transactions on Wireless Communications*, vol. 10, no. 4, pp. 1108–1120, Apr. 2011.

[55] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, “An analysis of approximations for maximizing submodular set functions - II,” *Mathematical Programming Study*, pp. 73–87, 1978.

[56] P. A. Forero, K. Rajawat, and G. B. Giannakis, “Prediction of partially observed dynamical processes over networks via dictionary learning,” in preparation.

[57] ——, “Semi-supervised dictionary learning for network-wide link load prediction,” in *Proc. Cognitive Information Processing Workshop*, Baiona, Spain, May 2012.

[58] C. Fragouli and A. Markopoulou, “A network coding approach to network monitoring,” in *Proc. of the 43rd Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sep. 2005.

[59] C. Fragouli, A. Markopoulou, and S. Diggavi, “Topology inference using network coding techniques,” in *Proc. of the 44th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sep. 2006.

[60] C. Fragouli and E. Soljanin, “Network coding applications,” *Foundations and Trends in Networking*, vol. 2, no. 2, pp. 135–269, 2007.

[61] ——, “Network coding fundamentals,” *Foundations and Trends in Networking*, vol. 2, no. 1, pp. 1–133, 2007.

[62] C. Fragouli, D. Katabi, A. Markopoulou, M. Médard, and H. Rahul, “Wireless network coding: Opportunities and challenges,” in *Proc. of the IEEE Military Communication Conference*, Orlando, FL, Oct. 2007, pp. 1–8.

[63] A. Frangioni and A. Manca, “A computational study of cost reoptimization for min-cost flow problems,” *INFORMS Journal on Computing*, vol. 18, no. 1, pp. 61–70, Winter 2006.

[64] N. Gatsis, A. Ribeiro, and G. Giannakis, “A class of convergent algorithms for resource allocation in wireless fading networks,” *IEEE Transactions on Wireless Communications*, vol. 9, no. 5, pp. 1808–1823, May 2010.

[65] L. Georgiadis, M. J. Neely, and L. Tassiulas, “Resource allocation and cross-layer control in wireless networks,” *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[66] C. Gkantsidis and P. R. Rodriguez, “Network coding for large scale content distribution,” in *Proc. of the IEEE INFOCOM*, vol. 4, Miami, FL, Mar. 2005, pp. 2235–2245.

[67] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[68] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[69] J. Haupt, W. U. Bajwa, M. Rabbat, and R. Nowak, “Compressed sensing for networked data,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 92–101, Mar. 2008.

[70] S. Haykin, “Cognitive radio: brain-empowered wireless communications,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, Feb. 2005.

[71] M. Heindlmaier, D. Traskov, R. Kötter, and M. Médard, “Scheduling for network coded multicast: A distributed approach,” in *Proc. of the IEEE GLOBECOM Workshops*, Honululu, HI, Nov. 2009, pp. 1–6.

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

[73] T. Ho, D. R. Karger, M. Médard, and R. Kötter, “Network coding from a network flow perspective,” in *Proc. of the IEEE International Symposium on Information Theory*, London, UK, May 2005.

[74] T. Ho and H. Viswanathan, “Dynamic algorithms for multicast with intra-session network coding,” *IEEE Transactions on Information Theory*, vol. 55, no. 2, pp. 797–815, Feb. 2009.

[75] C. Ibars, L. Giupponi, and S. Addepalli, “Distributed multiple access and flow control for wireless network coding,” in *Proc. of the IEEE Vehicular Technology Conference (VTC-Spring)*, Taipei, Taiwan, May 2010, pp. 1–6.

[76] M. Jafari, L. Keller, C. Fragouli, and K. Argyraki, “Compressed network coding vectors,” in *Proc. of the IEEE International Symposium on Information Theory*, Seoul, Korea, July 2009, pp. 109–113.

[77] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Médard, and M. Effros, “Resilient network coding in the presence of byzantine adversaries,” *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2596–2603, June 2008.

[78] 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 Transactions on Information Theory*, vol. 51, no. 6, pp. 1973–1982, June 2005.

[79] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, “XORs in the air: Practical wireless network coding,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, June 2008.

[80] S. Katti, D. Katabi, H. Balakrishnan, and M. Médard, “Symbol-level network coding for wireless mesh networks,” in *Proc. of the ACM SIGCOMM*, vol. 38, no. 4, Seattle, WA, Oct. 2008, pp. 401–412.

[81] D. I. Kim, L. B. Le, and E. Hossain, “Joint rate and power allocation for cognitive radios in dynamic spectrum access environment,” *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 5517–5527, Oct. 2008.

[82] S.-J. Kim, E. Dall’Anese, and G. B. Giannakis, “Cooperative spectrum sensing for cognitive radios using Kriged Kalman filtering,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 1, pp. 24–36, Feb. 2011.

[83] K. C. Kiwiel and P. O. Lindberg, “Parallel subgradient methods for convex optimization,” in *Inherently Parallel Algorithms in Feasibility and Optimization*, D. Butnariu, Y. Censor, and S. Reich, Eds. Amsterdam, Netherlands: Elsevier Science B.V., 2001, pp. 335–344.

[84] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. New York: Springer, 2009.

[85] R. Kötter and F. R. Kschischang, “Coding for errors and erasures in random network coding,” *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3579–3591, Aug. 2008.

[86] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[87] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, “Structural analysis of network traffic flows,” in *Proc. of the ACM SIGMETRICS*, New York, NY, 2004, pp. 61–72.

[88] H.-K. Lee and S.-L. Kim, “Network coded ALOHA for wireless multihop networks,” in *Proc. of the IEEE Wireless Communications and Networking Conference*, Budapest, Hungary, Apr. 2009, pp. 1–5.

[89] D. Li, X. Lin, W. Xu, Z. He, and J. Lin, “Rate control for network coding based multicast: a hierarchical decomposition approach,” in *Proc. of the 5th International Conference Wireless Communications and Mobile Computing*, June 2009, pp. 181–185.

[90] X. Li, C.-C. Wang, and X. Lin, “Throughput and delay analysis on uncoded and coded wireless broadcast with hard deadline constraints,” in *Proc. of the IEEE INFOCOM*, San Diego, CA, 14-19 2010, pp. 1–5.

[91] Z. Li and B. Li, “Efficient and distributed computation of maximum multicast rates,” in *Proc. of the IEEE INFOCOM*, Miami, FL, Mar. 2005, pp. 1618–1628.

[92] Y. Liao, P. Geurts, and G. Leduc, “Network distance prediction based on decentralized matrix factorization,” in *Proc. of the IFIP Networking*, Chennai, India, May 2010.

[93] X. Lin, N. B. Shroff, and R. Srikant, “A tutorial on cross-layer optimization in wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[94] D. S. Lun, T. Ho, N. Ratnakar, M. Médard, and R. Kötter, “Network coding in wireless networks,” in *Cooperation in Wireless Networks: Principles and Applications*, F. H. P. Fitzek and M. D. Katz, Eds. Springer, 2006, ch. 5.

[95] D. S. Lun, M. Médard, R. Kötter, and M. Effros, "On coding for reliable communication over packet networks," in *Proc. of the 42nd Annual Allerton Conference Communications, Control, and Computing*, Monticello, IL, Sep. 2004, pp. 20–29.

[96] ——, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3–20, Mar. 2008.

[97] D. S. Lun, N. Ratnakar, M. Médard, R. Kötter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2608–2623, June 2006.

[98] Z.-Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 1, pp. 57–73, Feb. 2008.

[99] M. M. Roughan, "A case study of the accuracy of snmp measurements," *JECE*, vol. 2010, pp. 1–7, Jan. 2010.

[100] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*, 1st ed. Cambridge University Press, 2003.

[101] G. Maierbacher, J. Barros, and M. Médard, "Practical source-network decoding," in *Proc. of the 6th International Symposium on Wireless Communication Systems*, Tuscany, Italy, Sep. 2009, pp. 283–287.

[102] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Supervised dictionary learning," *Advances in Neural Information Processing Systems*, pp. 1033–1040, 2008.

[103] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53 –69, Jan. 2008.

[104] K. V. Mardia, C. Goodall, E. J. Redfern, and F. J. Alonso, "The Kriged Kalman filter," *Test*, vol. 7, no. 2, pp. 217–285, Dec. 1998.

[105] B. R. Marks and G. P. Wright, "A general inner approximation algorithm for nonconvex mathematical programs," *Operations Research*, vol. 26, no. 4, pp. 681–683, July–Aug. 1978.

[106] S. Massoud Amin and B. F. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *IEEE Power and Energy Magazine*, vol. 3, no. 5, pp. 34–41, Sep.–Oct. 2005.

[107] R. Mehra, “On the identification of variances and adaptive Kalman filtering,” *IEEE Transactions on Automatic Control*, vol. 15, no. 2, pp. 175–184, Apr. 1970.

[108] N. Mehta, J. Wu, A. Molisch, and J. Zhang, “Approximating a sum of random variables with a lognormal,” *IEEE Transactions on Wireless Communications*, vol. 6, no. 7, pp. 2690–2699, July 2007.

[109] M. Minoux, “Accelerated greedy algorithms for maximizing submodular set functions,” in *Optimization Techniques*, ser. Lecture Notes in Control and Information Sciences, J. Stoer, Ed. Springer Berlin / Heidelberg, 1978, vol. 7, pp. 234–243.

[110] I. Mitola, J. and J. Maguire, G. Q., “Cognitive radio: making software radios more personal,” *IEEE Personal Communications Magazine*, vol. 6, no. 4, pp. 13–18, Aug. 1999.

[111] K. Myers and B. Tapley, “Adaptive sequential estimation with unknown noise statistics,” *IEEE Transactions on Automatic Control*, vol. 21, no. 4, pp. 520–523, Aug. 1976.

[112] A. Nedić and D. P. Bertsekas, “The effect of deterministic noise in subgradient methods,” *Mathematical Programming*, vol. 125, no. 1, pp. 75–99, 2009.

[113] A. Nedić and A. Ozdaglar, “Approximate primal solutions and rate analysis for dual subgradient methods,” *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2009.

[114] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functions - I,” *Mathematical Programming*, no. 1, pp. 265–294, Dec. 1978.

[115] U. D. of Energy, “The smart grid: an introduction,” Washington, DC, Sep. 2008.

[116] B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by V1?” *Vision Research*, vol. 37, no. 23, pp. 3311 –3325, 1997.

[117] F. A. Onat, I. Stojmenovic, and H. Yanikomeroglu, “Generating random graphs for simulation of wireless ad-hoc, actuator, and Internet networks,” *Pervasive and Mobile Computing (Elsevier)*, vol. 4, no. 5, pp. 597–615, 2008.

[118] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Médard, “Codecast: a network-coding-based ad hoc multicast protocol,” *IEEE Transactions on Wireless Communications*, vol. 13, no. 5, pp. 76–81, Oct. 2006.

[119] S. Pattem, B. Krishnamachari, and R. Govindan, “The impact of spatial correlation on routing with compression in wireless sensor networks,” in *Proc. of the 3rd International Symposium on Information Processing in Sensor Networks*, Berkeley, CA, June 2004, pp. 28–35.

[120] J. M. Peha, “Approaches to spectrum sharing,” *IEEE Communications Magazine*, vol. 43, no. 2, pp. 10–12, Feb. 2005.

[121] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2000.

[122] C. E. Perkins, *Ad Hoc Networking*. Addison Wesley Professional, Dec. 2000.

[123] S. S. Pradhan and K. Ramchandran, “Distributed source coding using syndromes (DISCUS): design and construction,” *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, Mar. 2003.

[124] M. G. Rabbat, M. A. T. Figueiredo, and R. D. Nowak, “Network inference from co-occurrences,” *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 4053–4068, Sep. 2008.

[125] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, “Self-taught learning: transfer learning from unlabeled data,” in *Proc. of the International Conference on Machine learning*, ser. ICML ’07, 2007, pp. 759–766.

[126] K. Rajawat, A. Cano, and G. B. Giannakis, “Network-compressive coding for wireless sensors with correlated data,” *IEEE Transactions on Wireless Communications*, to be published.

[127] K. Rajawat, E. Dall’Anese, and G. B. Giannakis, “Joint rate and power control for coded cognitive radio networks,” in *Proc. of 45th Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2011, pp. 259–264.

[128] ——, “Dynamic network kriging,” in *Proc. of the IEEE Statistical Signal Processing Workshop*, Ann Arbor, MI, Aug. 2012.

[129] ——, “Dynamic network delay cartography,” *IEEE Transactions on Information Theory*, submitted for publication.

[130] K. Rajawat, N. Gatsis, and G. B. Giannakis, “Cross-layer designs in coded wireless fading networks with multicast,” *IEEE/ACM Transactions on Networking*, vol. 19, no. 5, pp. 1276–1289, Oct. 2011.

[131] K. Rajawat, N. Gatsis, S.-J. Kim, and G. B. Giannakis, "Cross-layer design of coded multicast for wireless random access networks," *IEEE Transactions on Signal Processing*, vol. 29, no. 10, pp. 1970–1980, Dec. 2011.

[132] ——, "Cross-layer design of coded multicast for wireless random access networks," in *Proc. 45th Annual Conference Information Sciences and Systems*, Princeton, NJ, Mar. 2011, pp. 259–264.

[133] K. Rajawat and G. B. Giannakis, "Non-random wireless network coding," in *Proc. of the Second IEEE Workshop on Wireless Network Coding*, June 2009, pp. 1–6.

[134] ——, "Joint scheduling and network coding for multicast in delay-constrained wireless networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6186–6196, Dec. 2011.

[135] K. Rajawat, T. Wang, and G. B. Giannakis, "An algebraic polyphase approach to wireless network coding," in *Proc. of the IEEE International Conference Acoustics, Speech, and Signal Processing*, Taipei, Taiwan, Apr. 2009, pp. 2441–2444.

[136] K. Rajawat, N. Gatsis, and G. B. Giannakis, "Cross-layer design of multicast in fading: Network coding and asynchronous subgradients," in *Proc. of the Third IEEE Workshop on Wireless Network Coding*, June 2010, pp. 1–6.

[137] A. K. Ramasubramonian and J. W. Woods, "Multiple description coding and practical network coding for video multicast," *IEEE Signal Processing Letters*, vol. 17, no. 3, pp. 265–268, Mar. 2010.

[138] R. Rao and A. Ephremides, "On the stability of interacting queues in a multiple-access system," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 918–930, Sep. 1988.

[139] A. Ribeiro and G. B. Giannakis, "Separation theorems of wireless networking," *IEEE Transactions on Information Theory*, vol. 56, no. 9, Sep. 2010.

[140] M. Riemensberger, M. Heindlmaier, A. Dotzler, D. Traskov, and W. Utschick, "Optimal slotted random access in coded wireless packet networks," in *Proc. of the 6th Workshop Resource Allocation in Wireless Networks (RAWNET)*, Avignon, France, June 2010, pp. 374–379.

[141] B. D. Ripley, *Spatial Statistics*. John Wiley & Sons, 1981.

[142] S. M. Ross, *Introduction to Probability Models*, 8th ed. San Diego, CA: Academic Press, 2003.

[143] A. Ruszczyński, *Nonlinear Optimization*. Princeton, NJ: Princeton University Press, 2006.

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

[145] V. Saligrama, M. Alanyali, and O. Savas, “Distributed detection in sensor networks with packet losses and finite capacity links,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4118–4132, Nov. 2006.

[146] A. Scaglione and S. Servetto, “On the interdependence of routing and data compression in multi-hop sensor networks,” *Wireless Networks*, vol. 11, no. 1-2, pp. 149–160, Jan. 2005.

[147] T. Schmid, H. Dubois-Ferriere, and M. Vetterli, “Sensorscope: Experiences with a wireless building monitoring sensor network,” in *Proc. of the Workshop on Real-World Wireless Sensor Networks*, Stockholm, Sweden, June 2005, pp. 13–17.

[148] A. Schriver, *Combinatorial Optimization – Polyhedra and Efficiency*. Springer-Verlag, Berlin, Germany, 2003, vol. 1.

[149] H. Seferoglu and A. Markopoulou, “Opportunistic network coding for video streaming over wireless,” in *Proc. of the Packet Video Conference*, Lausanne, Switzerland, Nov. 2007, pp. 191–200.

[150] S. Shakkottai and R. Srikant, “Network optimization and control,” *Foundations and Trends in Networking*, vol. 2, no. 3, pp. 271–379, 2007.

[151] Y. Shavitt, X. Sun, A. Wool, and B. Yener, “Computing the unmeasured: an algebraic approach to Internet mapping,” *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 67–78, Jan. 2004.

[152] D. Silva and F. Kschischang, “Rank-metric codes for priority encoding transmission with network coding,” in *Proc. of the Canadian Workshop on Information Theory*, Edmonton, AB, June 2007, pp. 81–84.

[153] H. Singhal and G. Michailidis, “Structural models for dual modality data with application to network tomography,” *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5054–5071, Aug. 2011.

[154] R. Sinkhorn, “A relationship between arbitrary positive matrices and doubly stochastic matrices,” *The Annals of Mathematical Statistics*, vol. 35, no. 2, pp. 876–879, 1964.

[155] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, “Traffic matrices: balancing measurements, inference and modeling,” *SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 362–373, June 2005.

[156] A. Swami, Q. Zhao, Y. Hong, and L. Tong, *Wireless Sensor Networks: Signal Processing and Communications Perspective*. Wiley, 2007.

[157] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[158] I. Toνsić and P. Frossard, “Dictionary learning,” *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[159] D. Traskov, M. Heindlmaier, M. Médard, R. Kötter, and D. S. Lun, “Scheduling for network coded multicast: A conflict graph formulation,” in *Proc. of the IEEE GLOBECOM Workshops*, New Orleans, LA, Nov.–Dec. 2008, pp. 1–5.

[160] D. Traskov, D. S. Lun, R. Kötter, and M. Médard, “Network coding in wireless networks with random access,” in *Proc. of the IEEE International Symposium Information Theory (ISIT)*, Nice, France, June 2007, pp. 2726–2730.

[161] D. Umehara, T. Hirano, S. Denno, M. Morikura, and T. Sugiyama, “Wireless network coding in slotted ALOHA with two-hop unbalanced traffic,” *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 5, pp. 647–661, June 2009.

[162] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, “Tree-based reparameterization framework for analysis of sum-product and related algorithms,” *IEEE Transactions on Information Theory*, vol. 49, no. 5, pp. 1120–1146, May 2003.

[163] M. J. Wainwright and M. I. Jordan, “Graphical models, exponential families, and variational inference,” *Foundations and Trends in Machine Learning*, vol. 1, no. 1 and 2, pp. 1–305, 2008.

[164] D. Wang, Q. Zhang, and J. Liu, “Partial network coding: Concept, performance, and application for continuous data collection in sensor networks,” *ACM Transactions on Sensor Networks*, vol. 4, no. 3, pp. 1–22, May 2008.

[165] R. Wang, V. Lau, L. Linjun, and B. Chen, “Joint cross-layer scheduling and spectrum sensing for ofdma cognitive radio systems,” *IEEE Transactions on Wireless Communications*, vol. 8, no. 5, pp. 2410–2416, May 2009.

[166] C. K. Wikle and N. Cressie, “A dimension-reduced approach to space-time Kalman filtering,” *Biometrika*, vol. 86, no. 4, pp. 815–829, 1999.

[167] L. A. Wolsey, *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, Sep. 1998.

[168] Y. Wu, M. Chiang, and S.-Y. Kung, “Distributed utility maximization for network coding based multicasting: A critical cut approach,” in *Proc. of the 4th International Symposium Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, Boston, MA, Apr. 2006, pp. 1–6.

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

[170] Y. Xi and E. Yeh, “Distributed algorithms for minimum cost multicast with network coding,” *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 379–392, 2010.

[171] Z. Xiong, A. D. Liveris, and S. Cheng, “Distributed source coding for sensor networks,” *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–94, Sep. 2004.

[172] W. Xu, E. Mallada, and A. Tang, “Compressive sensing over graphs,” in *Proc. of the IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 2087–2095.

[173] X. Yan, M. J. Neely, and Z. Zhang, “Multicasting in time-varying wireless networks: Cross-layer dynamic resource allocation,” in *IEEE International Symposium Information Theory*, Nice, France, June 2007, pp. 2721–2725.

[174] W. Ye, J. Heidemann, and D. Estrin, “An energy-efficient mac protocol for wireless sensor networks,” in *Proc. of the IEEE Conference on Computer Communications*, New York, USA, June 2002, pp. 1567–1576.

[175] R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, “Network coding theory,” *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 4 and 5, pp. 241–381, 2005.

[176] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, “Fast accurate computation of large-scale IP traffic matrices from link loads,” *SIGMETRICS Performance Evaluation Review*, vol. 31, pp. 206–217, June 2003.

[177] Y. Zhang, M. Roughan, C. Lund, and D. L. Donoho, “Estimating point-to-point and point-to-multipoint traffic matrices: an information-theoretic approach,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 947 – 960, Oct. 2005.

[178] Z. Zhang, “Linear network error correction codes in packet networks,” *IEEE Transactions on Information Theory*, vol. 54, no. 1, pp. 209–218, Jan. 2008.

[179] Q. Zhao and B. M. Sadler, “A survey of dynamic spectrum access,” *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 79–89, May 2007.

[180] H. Zhu, G. B. Giannakis, and A. Cano, “Distributed in-network channel decoding,” *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3970–3983, Oct. 2009.