A Linear Network Code Construction for General Integer Connections Based on the Constraint Satisfaction Problem

Ying Cui, Muriel Médard, Fan Lai, Edmund Yeh, Douglas Leith, Ken Duffy, Dhaivat Pandya

***Abstract*— The problem of finding network codes for general connections is inherently difficult in capacity constrained networks. Resource minimization for general connections with network coding is further complicated. Existing methods for identifying solutions mainly rely on highly restricted classes of network codes, and are almost all centralized. In this paper, we introduce linear network mixing coefficients for code constructions of general connections that generalize random linear network coding (RLNC) for multicast connections. For such code constructions, we pose the problem of cost minimization for the subgraph involved in the coding solution and relate this minimization to a path-based Constraint Satisfaction Problem (CSP) and an edge-based CSP. While CSPs are NP-complete in general, we present a path-based probabilistic distributed algorithm and an edge-based probabilistic distributed algorithm with almost sure convergence in finite time by applying Communication Free Learning (CFL). Our approach allows fairly general coding across flows, guarantees no greater cost than routing, and shows a possible distributed implementation. Numerical results illustrate the performance improvement of our approach over existing methods.**

***Index Terms*— network coding, network mixing, general connection, resource optimization, distributed algorithm.**

I. Introduction

The problem of finding network codes in the case of general connections, where each destination can request information from any subset of sources, is intrinsically difficult and little is known about its complexity. In certain special cases, such as multicast connections (where destinations share all of their demands), it suffices to satisfy a Ford-Fulkerson type of min-cut max-flow constraint between all sources to every destination individually. For multicast connections, linear codes suffice [1], [2], and lend themselves to a distributed random construction [3]. While linear codes have been the most widely considered in the literature, linear codes over finite fields may in general not be sufficient for general connections, as shown by [4] using an example from matroid theory.

A matroidal structure for the network coding problem with general connections was conjectured by the late Ralf Kötter (private communication) but, while different aspects of this connection have been investigated in the literature [5]–[11], a proof remains elusive, except in special cases. Recently, the problem of scalar-linear coding has been shown to have a matroidal structure [7], [8], [12]. There exists a correspondence between scalar-linearly solvable networks and representable matroids over finite fields, which can be used to obtain some bounds on scalar linear network capacity [13] or the capacity regions of certain classes of networks [14]. More generally, the problem of finding the linear network coding capacity region is equivalent to the characterization of all linear polymatroids [9], whose structure was investigated in [10]. Reference [11] generalized the results of [15], which investigated the connection among index coding, network coding and matroid theory. In [16], polymatroids were used to produce linear code constructions.

Progress in understanding the matroidal structure of the general connection problem has, however, not yet provided simple and useful approaches to generating explicit linear codes. There has been considerable investigation of restricted cases, such as a network with only two sources and two destinations, generally referred to as the two-unicast network [17]–[21], but thus far such investigation has yielded only bounds or linear solutions for restricted cases of the two-unicast network. It has been shown in [20] that the two-unicast problem is as hard as the most general network coding problem. Since the difficulty of coding in the case of general connections is in effect an interference cancellation one, approaches relying on interference alignment have naturally been explored [22]–[24]. Reference [25] investigated the enumeration, rate region computation and hierarchy of general multi-source multi-sink hyper-edge networks under network coding.

Even when we consider simple scalar network codes, which have scalar coding coefficients, the problem of code construction for general connections remains vexing. The main difficulty lies in cancelling the effect of flows that are coded together even though they are not destined for a common destination. The problem of code construction is further complicated when we seek, for common reasons of network resource management, to limit fully or partially the use of links in the network. For convex cost functions of flows over edges in the graph corresponding to the network, finding a minimum-cost solution is known to be a convex optimization problem in the case of multicast connections (for continuous flows) [26]. However, in the case of general connections, network resource minimization, even when allowing only restricted code constructions, appears difficult.

Among coding approaches for optimizing network use for general connections, we distinguish two types. The first, which we adopt in this paper, is that of mixing, by which we mean

Y. Cui and F. Lai are with Shanghai Jiao Tong University. M. Médard is with MIT. E. Yeh is with Northeastern University. D. Leith is with Trinity College Dublin. K. Duffy is with Maynooth University. D. Pandya is with Harvard University. The paper was presented in part in GLOBECOM 2015.

coding together flows using random linear network coding (RLNC) [3], originally proposed for multicast connections. The principle is to code together flows as though they were part of a common multicast connection. In this case, no explicit coding coefficients are provided, and decidability is ensured with high probability by RLNC. For example, the mixing approaches in [27] and [28] are both based on mixing variables, each corresponding to a set of flows that can be mixed over an edge. Specifically, in [27], a two-step mixing approach is proposed for network resource minimization of general connections, where flow partition (mixing) and flow rate optimization are considered separately. This separation imposes stronger restrictions on the mixing design in the first step and leads to a limitation on the feasibility region. Reference [28] studies the feasibility of more general mixing designs based on mixing variables of size $\mathcal{O}(2^P)$, where P is the number of flows. Reference [28] does not, however, provide an approach for obtaining a specific mixing design. The second type of coding approach is an explicit linear code construction, by which we mean providing specific linear coefficients over some finite field, to be applied to coding flows at different nodes. Often these constructions are simplified by restricting them to be pairwise. For example, in [29] and [30], simple codes over pairs of flows are proposed for network resource minimization of general connections.

Some explicit linear network code construction approaches [29], [30] are distributed, but they allow only pairwise coding. The algorithms of [31] using evolutionary techniques, which are also explicit code constructions, are partially distributed, since the chromosomes can be decomposed into their local contributions, but require information to be fed back from the receivers to all the nodes in the network. In addition, the convergence results for evolutionary techniques are generally scant and do not yield prescriptive constructions. While RLNC for multicast connections is a distributed algorithm, most of the mixing approaches [27], [28] based on it have remained centralized. In [32], we propose new methods for constructing linear network codes for general connections of continuous flows based on mixing to minimize the total network cost. Flow splitting and coding over time are required to achieve the desired performance. The focus in [32] is to apply continuous optimization techniques to obtain continuous flow rates. In [33], we consider linear network code construction for general connections of integer flows based on mixing, and propose an edge-based probabilistic distributed algorithm to minimize the total network cost. This paper extends the results in [33].

Our contribution in this paper is to present new methods for constructing linear network codes in a distributed manner for general connections of integer flows based on mixing.

• We introduce linear network mixing coefficients. The number of mixing coefficients grows polynomially with the number of flows. We formally establish the relationship between linear network coding and mixing.

• We formulate the minimization of the cost of the subgraph involved in the code construction for general connections of integer flows in terms of the mixing coefficients.

• We relate our problem to a path-based Constraint Satisfaction Problem (CSP) and an edge-based CSP. While CSPs are NP-complete in general, we present a path-based probabilistic distributed algorithm and an edge-based probabilistic distributed algorithm with almost sure convergence in finite time by applying Communication Free Learning (CFL), a recent probabilistic distributed solution for CSPs [34]. The path-based distributed algorithm requires more local information than the edge-based distributed algorithm, but converges faster.

• We show that our approach guarantees no greater cost than routing or the simplified mixing design in [27]. Numerical results also illustrate the performance improvement of our approach over existing methods.

While our approach, like all other general connection code constructions, is generally suboptimal, it allows more flows to be mixed than is possible with pairwise mixing [29], [30] and with the separate mixing design in [27]. Moreover, in contrast to [27], [28], [32], our approach does not require non-scalar coding over time.

II. Problem Setup and Definitions

A. Network Model

We consider a directed acyclic network with general connections.[1] Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the directed acyclic graph, where $\mathcal{V}$ denotes the set of $V = |\mathcal{V}|$ nodes and $\mathcal{E}$ denotes the set of $E = |\mathcal{E}|$ edges. To simplify notation, we assume there is only one edge from node $i \in \mathcal{V}$ to node $j \in \mathcal{V}$, denoted as edge $(i, j) \in \mathcal{E}$.[2] For each node $i \in \mathcal{V}$, define the set of incoming neighbors to be $\mathcal{I}_i = \{j : (j, i) \in \mathcal{E}\}$ and the set of outgoing neighbors to be $\mathcal{O}_i = \{j : (i, j) \in \mathcal{E}\}$. Let $I_i = |\mathcal{I}_i|$ and $O_i = |\mathcal{O}_i|$ denote the in-degree and out-degree of node $i \in \mathcal{V}$, respectively. Assume $I_i \leq D$ and $O_i \leq D$ for all $i \in \mathcal{V}$, where D is a constant.

Consider a finite field $\mathcal{F}$ with size $F = |\mathcal{F}|$. Let $\mathcal{P} = \{1, \cdots, P\}$ denote the set of $P = |\mathcal{P}|$ flows of symbols in finite field $\mathcal{F}$ to be carried by the network. For each flow $p \in \mathcal{P}$, let $s_p \in \mathcal{V}$ be its source. We consider integer flows. To simplify notation, we assume unit source rate (i.e., one finite field symbol per second).[3] Let $\mathcal{S} = \{s_1, \cdots, s_P\}$ denote the set of $P = |\mathcal{S}|$ sources. We assume different flows do not share a common source node and no source node has any incoming edges. Let $\mathcal{T} = \{t_1, \cdots, t_T\}$ denote the set of $T = |\mathcal{T}|$ terminals. Each terminal $t \in \mathcal{T}$ demands a subset of $P_t = |\mathcal{P}_t|$ flows $\mathcal{P}_t \subseteq \mathcal{P}$. Assume $\cup_{t\in\mathcal{T}} \mathcal{P}_t = \mathcal{P}$. Let $\boldsymbol{\mathcal{P}} \triangleq (\mathcal{P}_t)_{t\in\mathcal{T}}$ denote the demands of all the terminals. We assume no terminal has any outgoing edges.

As we consider integer flows, we assume unit edge capacity (i.e., one finite field symbol per second).[4] Let $z_{ij} \in \{0, 1\}$ denote whether edge $(i, j) \in \mathcal{E}$ is in the subgraph involved in

[1]The network model we considered in this paper is similar to that in [32] for continuous flows, but here we consider integer flows and edge capacities, and do not allow flow splitting and coding over time.

[2]Multiple edges from node i to node j can be modeled by introducing multiple extra nodes, one on each edge, to transform a multigraph intro a graph.

[3]A source with a positive integer source rate greater than one can be modeled by multiple sources, each with unit source rate.

[4]An edge with a positive integer edge capacity greater than one can be equivalently converted to multiple edges, each with unit edge capacity.

the code construction in a sense we shall make precise later.[5] We assume a cost is incurred on an edge when information is transmitted through the edge and let $U_{ij}(z_{ij})$ denote the cost function for edge (i,j). We assume $U_{ij}(z_{ij})$ is non-decreasing in z_{ij}. We are interested in the problem of finding linear network coding designs and minimizing the network cost $\sum_{(i,j)\in\mathcal{E}} U_{ij}(z_{ij})$ for general connections under those designs.

B. Scalar Time-Invariant Linear Network Coding

In linear network coding, a linear combination over $\mathcal{F}$ of the symbols in $\{\sigma_{ki} \in \mathcal{F} : k \in \mathcal{I}_i\}$ from the incoming edges $\{(k,i) : k \in \mathcal{I}_i\}$ can be transmitted through the shared edge $(i,j) \in \mathcal{E}$. The coefficients used to form this linear combination are referred to as local coding coefficients. Specifically, let $\alpha_{kij} \in \mathcal{F}$ denote the local coding coefficient corresponding to edge $(k,i) \in \mathcal{E}$ and edge $(i,j) \in \mathcal{E}$. Denote $\boldsymbol{\alpha} \triangleq (\alpha_{kij})_{(k,i),(i,j)\in\mathcal{E}}$. Then, for linear network coding, using local coding coefficients, the symbol through edge $(i,j) \in \mathcal{E}$ can be expressed as

$$\sigma_{ij} = \sum_{k\in\mathcal{I}_i} \alpha_{kij}\sigma_{ki}, \quad (i,j) \in \mathcal{E},\ i \notin \mathcal{S}. \tag{1}$$

This is illustrated in Fig. 1.

Starting from the sources, we transmit source symbols $\{\sigma_p \in \mathcal{F} : p \in \mathcal{P}\}$, and then, at intermediate nodes, we perform only linear operations over $\mathcal{F}$ on the symbols from incoming edges. Thus, the symbol of each edge can be expressed as a linear combination over $\mathcal{F}$ of the source symbols $\{\sigma_p \in \mathcal{F} : p \in \mathcal{P}\}$. Let $c_{ij,p} \in \mathcal{F}$ denote the coefficient of flow $p \in \mathcal{P}$ in the linear combination for edge $(i,j) \in \mathcal{E}$. This is referred to as the global coding coefficient of flow $p \in \mathcal{P}$ and edge $(i,j) \in \mathcal{E}$. Let $\mathbf{c}_{ij} \triangleq (c_{ij,1}, \cdots, c_{ij,p}, \cdots, c_{ij,P}) \in \mathcal{F}^P$ denote P coefficients corresponding to this linear combination for edge $(i,j) \in \mathcal{E}$. This is referred to as the global coding vector of edge $(i,j) \in \mathcal{E}$. Here, $\mathcal{F}^P$ represents the set of global coding vectors, the cardinality of which is F^P. Then, using global coding vectors, the symbol through edge $(i,j) \in \mathcal{E}$ can also be expressed as

$$\sigma_{ij} = \sum_{p\in\mathcal{P}} c_{ij,p}\sigma_p, \quad (i,j) \in \mathcal{E},\ i \notin \mathcal{S}. \tag{2}$$

This is illustrated in Fig. 1.

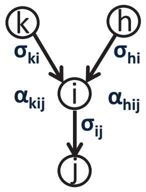


Fig. 1: Illustration of local and global coding coefficients. $\mathcal{P} = \{1,2\}$. Then, we have $\sigma_{ki} = c_{ki,1}\sigma_1 + c_{ki,2}\sigma_2$, $\sigma_{hi} = c_{hi,1}\sigma_1 + c_{hi,2}\sigma_2$, $\sigma_{ij} = \alpha_{kij}\sigma_{ki} + \alpha_{hij}\sigma_{hi} = c_{ij,1}\sigma_1 + c_{ij,2}\sigma_2$, $c_{ij,1} = \alpha_{kij}c_{ki,1} + \alpha_{hij}c_{hi,1}$ and $c_{ij,2} = \alpha_{kij}c_{ki,2} + \alpha_{hij}c_{hi,2}$.

[5]There is either no flow or a unit rate of (coded) flow through each edge. Under the unit source rate and edge capacity assumptions, we shall see that there is one global coding (mixing) vector for each edge.

In this paper, we consider scalar time-invariant linear network coding. In other words, $\alpha_{kij} \in \mathcal{F}$ and $c_{ij,p} \in \mathcal{F}$ are both scalars, and do not change over time. Let $\mathbf{e}_p$ denote the vector with the p-th element being 1 and all the other elements being 0. For decodability to hold at all the terminals, the global coding vectors at all edges must satisfy the following feasibility condition for scalar linear network coding.

Definition 1 (Feasibility of Scalar Linear Network Coding): For a network $\mathcal{G} = (\mathcal{V},\mathcal{E})$ and a set of flows $\mathcal{P}$ with sources $\mathcal{S}$ and terminals $\mathcal{T}$, a linear network code $\mathbf{c} \triangleq (\mathbf{c}_{ij})_{(i,j)\in\mathcal{E}}$ is called feasible if the following three conditions are satisfied: 1) $\mathbf{c}_{s_pj} = \mathbf{e}_p$ for source edge $(s_p,j) \in \mathcal{E}$, where $s_p \in \mathcal{S}$ and $p \in \mathcal{P}$; 2) $\mathbf{c}_{ij} = \sum_{k\in\mathcal{I}_i} \alpha_{kij}\mathbf{c}_{ki}$ for edge $(i,j) \in \mathcal{E}$ not outgoing from a source, where $i \notin \mathcal{S}$ and $\alpha_{kij} \in \mathcal{F}$; 3) $\mathbf{e}_p \in \text{span}\{\mathbf{c}_{it} : i \in \mathcal{I}_t\}$, where $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$.

Note that when using scalar linear network coding, for each terminal, extraneous flows are allowed to be mixed with the desired flows on the paths to the terminal, as the extraneous flows can be cancelled at intermediate nodes or at the terminal.

C. Scalar Time-Invariant Linear Network Mixing

As mentioned in Section I, to facilitate distributed linear network code designs for general connections using the mixing concept (without requiring the specific values of local or global coding coefficients in the designs), we introduce local and global mixing variables. Later, we shall see that distributed linear network mixing designs in terms of these mixing coefficients are much easier. Specifically, we introduce the local mixing coefficient $\beta_{kij} \in \{0,1\}$ corresponding to edge $(k,i) \in \mathcal{E}$ and edge $(i,j) \in \mathcal{E}$, which relates to the local coding coefficient $\alpha_{kij} \in \mathcal{F}$. Denote $\boldsymbol{\beta} \triangleq (\beta_{kij})_{(k,i),(i,j)\in\mathcal{E}}$. $\beta_{kij} = 1$ indicates that symbol σ_{ki} of edge $(k,i) \in \mathcal{E}$ is allowed (under our construction) to contribute to the linear combination over $\mathcal{F}$ forming symbol σ_{ij} in (1) and $\beta_{kij} = 0$ otherwise. Thus, if $\beta_{kij} = 0$, we have $\alpha_{kij} = 0$; if $\beta_{kij} = 1$, we can further determine how symbol σ_{ki} contributes to the linear combination forming symbol σ_{ij} by choosing $\alpha_{kij} \in \mathcal{F}$ (note that α_{kij} can be zero when $\beta_{kij} = 1$).

Similarly, we introduce the global mixing coefficient $x_{ij,p} \in \{0,1\}$ of flow $p \in \mathcal{P}$ and edge $(i,j) \in \mathcal{E}$, which relates to the global coding coefficient $c_{ij,p} \in \mathcal{F}$. $x_{ij,p} = 1$ indicates that flow p is allowed (under our construction) to be mixed (coded) with other flows, i.e., symbol σ_p is allowed to contribute to the linear combination over $\mathcal{F}$ forming symbol σ_{ij} in (2), and $x_{ij,p} = 0$ otherwise. Thus, if $x_{ij,p} = 0$, we have $c_{ij,p} = 0$; if $x_{ij,p} = 1$, we can further determine how symbol σ_p contributes to the linear combination forming symbol σ_{ij} (note that $c_{ij,p}$ can be zero when $x_{ij,p} = 1$). Then, we introduce the global mixing vector $\mathbf{x}_{ij} \triangleq (x_{ij,1}, \cdots, x_{ij,p}, \cdots, x_{ij,P}) \in \{0,1\}^P$ for edge $(i,j) \in \mathcal{E}$, which relates to the global coding vector $\mathbf{c}_{ij} = (c_{ij,1}, \cdots, c_{ij,p}, \cdots, c_{ij,P}) \in \mathcal{F}^P$. Here, $\{0,1\}^P$ represents the set of global mixing vectors, the cardinality of which is 2^P.

We consider scalar time-invariant linear network mixing. In other words, $\beta_{kij} \in \{0,1\}$ and $x_{ij,p} \in \{0,1\}$ are both scalars, and β_{kij} and $x_{ij,p}$ do not change over time.

Global mixing vectors provide a natural way of speaking of flows as possibly coded or not without knowledge of the specific values of global coding vectors. Intuitively, global mixing vectors can be regarded as a limited representation of global coding vectors. Given network mixing vectors, it may not be sufficient to tell whether a certain symbol can be decoded or not. Thus, using the network mixing representation, the extraneous flows, when mixed with the desired flows on the paths to each terminal, are not guaranteed to be cancelled at the terminal. For decodability to hold at all the terminals, the global mixing vectors at all edges must satisfy the following feasibility condition for scalar linear network mixing.

Definition 2 (Feasibility of Scalar Linear Network Mixing): For a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set of flows $\mathcal{P}$ with sources $\mathcal{S}$ and terminals $\mathcal{T}$, a linear network mixing design $\mathbf{x} \triangleq (\mathbf{x}_{ij})_{(i,j)\in\mathcal{E}}$ is called feasible if the following three conditions are satisfied: 1) $\mathbf{x}_{s_p j} = \mathbf{e}_p$ for source edge $(s_p, j) \in \mathcal{E}$, where $s_p \in \mathcal{S}$ and $p \in \mathcal{P}$; 2) $\mathbf{x}_{ij} = \vee_{k\in\mathcal{I}_i} \beta_{kij}\mathbf{x}_{ki}$ for edge $(i,j) \in \mathcal{E}$ not outgoing from a source, where $i \notin \mathcal{S}$ and $\beta_{kij} \in \{0,1\}$;[6] 3) $x_{it,p} = 0$, where $i \in \mathcal{I}_t$, $p \notin \mathcal{P}_t$, $t \in \mathcal{T}$.

Note that Condition 3) in Definition 2 ensures that for each terminal, the extraneous flows are not mixed with the desired flows on the paths to the terminal. In other words, linear mixing allows only mixing at intermediate nodes. This is not as general as using linear network coding, which allows mixing and canceling (i.e., removing one or multiple flows from a mixing of flows) at intermediate nodes.

Given a feasible linear network mixing design, one of the ways to implement mixing when $\mathcal{F}$ is large is to use random linear network coding (RLNC) [3], [27], as discussed in the introduction. In particular, when $\beta_{kij} = 1$, α_{kij} can be randomly, uniformly, and independently chosen in $\mathcal{F}$ using RLNC; when $\beta_{kij} = 0$, α_{kij} has to be chosen to be 0.

III. Mixing Problem Formulation

In this section, we formulate the problem of selecting mixing coefficients $\boldsymbol{\beta}$ and $\mathbf{x}$ to minimize the cost of the subgraph involved in the coding solution, i.e., the set of edges used in delivering the flows.

In the following formulation, $z_{ij} \in \{0,1\}$ indicates whether edge $(i,j) \in \mathcal{E}$ is involved in delivering flows, and $f^t_{ij,p} \in \{0,1\}$ indicates whether edge $(i,j) \in \mathcal{E}$ is involved in delivering flow $p \in \mathcal{P}_t$ to terminal $t \in \mathcal{T}$.

Problem 1 (Mixing):

$$U^*(\boldsymbol{\mathcal{P}}) \triangleq \min_{\mathbf{z},\mathbf{f},\mathbf{x},\boldsymbol{\beta}} \sum_{(i,j)\in\mathcal{E}} U_{ij}(z_{ij})$$

$$s.t.\ z_{ij} \in \{0,1\},\ (i,j)\in\mathcal{E} \tag{3}$$

$$x_{ij,p} \in \{0,1\},\ (i,j)\in\mathcal{E},\ p\in\mathcal{P} \tag{4}$$

$$\beta_{kij} \in \{0,1\},\ (k,i),(i,j)\in\mathcal{E} \tag{5}$$

$$f^t_{ij,p} \in \{0,1\},\ (i,j)\in\mathcal{E},\ p\in\mathcal{P}_t,\ t\in\mathcal{T} \tag{6}$$

$$\sum_{p\in\mathcal{P}_t} f^t_{ij,p} \le z_{ij},\ (i,j)\in\mathcal{E},\ t\in\mathcal{T} \tag{7}$$

$$\sum_{k\in\mathcal{O}_i} f^t_{ik,p} - \sum_{k\in\mathcal{I}_i} f^t_{ki,p} = \sigma^t_{i,p},\ i\in\mathcal{V},\ p\in\mathcal{P}_t,\ t\in\mathcal{T} \tag{8}$$

$$f^t_{ij,p} \le x_{ij,p},\ (i,j)\in\mathcal{E},\ p\in\mathcal{P}_t,\ t\in\mathcal{T} \tag{9}$$

$$\mathbf{x}_{s_p j} = \mathbf{e}_p,\ (s_p,j)\in\mathcal{E},\ p\in\mathcal{P} \tag{10}$$

$$\mathbf{x}_{ij} = \vee_{k\in\mathcal{I}_i}\beta_{kij}\mathbf{x}_{ki},\ (i,j)\in\mathcal{E},\ i\notin\mathcal{S} \tag{11}$$

$$x_{it,p} = 0,\ i\in\mathcal{I}_t,\ p\notin\mathcal{P}_t,\ t\in\mathcal{T} \tag{12}$$

where

$$\sigma^t_{i,p} = \begin{cases} 1, & i = s_p \\ -1, & i = t \\ 0, & \text{otherwise} \end{cases} \quad i\in\mathcal{V},\ p\in\mathcal{P}_t,\ t\in\mathcal{T}. \tag{13}$$

Here, $\mathbf{z} \triangleq (z_{ij})_{(i,j)\in\mathcal{E}}$ and $\mathbf{f} \triangleq (f^t_{ij,p})_{(i,j)\in\mathcal{E},p\in\mathcal{P}_t,t\in\mathcal{T}}$.[7]

Consider a feasible solution $\mathbf{z}$, $\mathbf{f}$, $\mathbf{x}$ and $\boldsymbol{\beta}$ to Problem 1. By (6) and (8), we know that for all $p\in\mathcal{P}_t$ and $t\in\mathcal{T}$, all the edges in $\{(i,j)\in\mathcal{E} : f^t_{ij,p} = 1\}$ form one flow path (i.e., a set of ordered edges $(i,j)\in\mathcal{E}$ such that $f^t_{ij,p} = 1$) from source s_p to terminal t. In addition, combining (3) and (7), we have an equivalent constraint purely in terms of $\mathbf{f}$, i.e.,

$$\sum_{p\in\mathcal{P}_t} f^t_{ij,p} \in \{0,1\}, \quad (i,j)\in\mathcal{E},\ t\in\mathcal{T}. \tag{14}$$

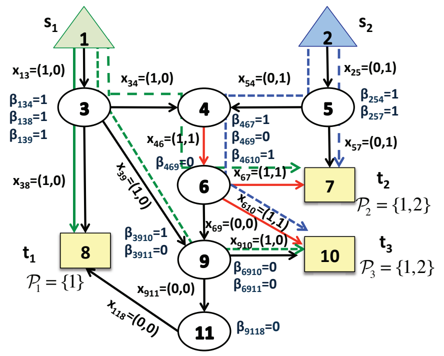
From this, we know that for all $p, p' \in \mathcal{P}_t$, $p \ne p'$ and $t\in\mathcal{T}$, the two flow paths from sources s_p and $s_{p'}$ to terminal t are edge-disjoint. Finally, the feasibility constraints in (10), (11) and (12) together with (4) and (5) set other requirements on flow paths (i.e., $\mathbf{f}$) via the constraint in (9). Therefore, a feasible solution to Problem 1 corresponds to a set of flow paths satisfying certain requirements, as illustrated above. These interpretations can be understood from the following example.

Example 1 (Illustration of Problem 1): As illustrated in Fig. 2, we consider a network with $\mathcal{P} = \{1,2\}$, $\mathcal{S} = \{1,2\}$, $\mathcal{T} = \{8,7,10\}$, $\mathcal{P}_1 = \{1\}$, $\mathcal{P}_2 = \mathcal{P}_3 = \{1,2\}$, and $U_{ij}(z_{ij}) = z_{ij} \in \{0,1\}$ for all $(i,j)\in\mathcal{E}$. Problem 1 for the network in Fig. 2 has two feasible solutions of network costs 11 and 12, as illustrated in Fig. 2 (a) and Fig. 2 (b), respectively. Specifically, the two feasible solutions share the same flow paths from sources s_1 and s_2 to terminals t_2 and t_3, i.e., flow paths $1-3-4-6-7$, $2-5-7$, $1-3-9-10$, $2-5-4-6-10$. Notice that the two feasible solutions have different flow paths from source s_1 to terminal t_1, i.e., flow path $1-3-8$ for the feasible solution in Fig. 2 (a) and flow path $1-3-9-11-8$ for the feasible solution in Fig. 2 (b). The optimal solution is the one illustrated in Fig. 2 (a) and the optimal network cost is 11.

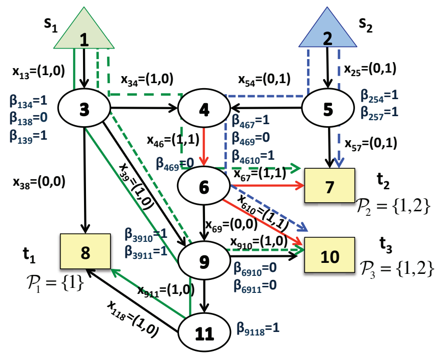
We now illustrate the complexity of Problem 1. The number of variables in $\boldsymbol{\beta}$ is $\sum_{(i,j)\in\mathcal{E}} O_j = \sum_{j\in\mathcal{V}} I_j O_j \le D\sum_{j\in\mathcal{V}} O_j = DE$. The number of variables in $\mathbf{f}$ is smaller than or equal to PTE. The numbers of variables in $\mathbf{z}$ and $\mathbf{x}$ are E and PE, respectively. Therefore, the total number of variables in Problem 1 is smaller than or equal to $(D+1)E + (T+1)PE$, i.e., polynomial in E, T and P. Problem 1 is a binary optimization problem, and does not appear to have a ready solution.

[6]Note that $\vee$ denotes the "or" operator (logical disjunction).

[7]Note that the optimal value in Problem 1 is a function of $\mathcal{G}$, $\mathcal{S}$, $\mathcal{T}$ and $\boldsymbol{\mathcal{P}}$, which are assumed to be fixed. Here, we write $U^*(\boldsymbol{\mathcal{P}})$ as a function of $\boldsymbol{\mathcal{P}}$ only to emphasize the impact of $\boldsymbol{\mathcal{P}}$ on $U^*(\boldsymbol{\mathcal{P}})$, which is helpful when considering demand set expansion in Problem 2.



(a) One feasible solution of network cost 11.



(b) One feasible solution of network cost 12.

Fig. 2: Illustration of feasible solutions to Problem 1. $\mathcal{P} = \{1, 2\}$, $\mathcal{S} = \{1, 2\}$, $\mathcal{T} = \{8, 7, 10\}$, $\mathcal{P}_1 = \{1\}$, $\mathcal{P}_2 = \mathcal{P}_3 = \{1, 2\}$, $U_{ij}(z_{ij}) = z_{ij} \in \{0, 1\}$ for all $(i, j) \in \mathcal{E}$. The flow paths from the two sources are illustrated using green and blue curves, respectively. Since $\mathcal{P}_2 = \mathcal{P}_3 = \{1, 2\}$, the flows from s_1 to t_2 and s_2 to t_3 are allowed to be mixed at edge $(4, 6)$. The red edges carry network-coded information. In this topology, Problem 1 has two feasible solutions. However, neither the two-step mixing approach for general connections in [27] nor routing provides a feasible solution.

Remark 1 (Problem 1 for Multicast): When $\mathcal{P}_t = \mathcal{P}$ for all $t \in \mathcal{T}$ (i.e., multicast), the constraint in (12) does not exist, and the constraint in (9) is always satisfied by choosing $\beta_{kij} = 1$ for all $(k, i), (i, j) \in \mathcal{E}$ and choosing $\mathbf{x}$ accordingly by (10) and (11). Therefore, Problem 1 for general connections reduces to the conventional minimum-cost scalar time-invariant linear network code design problem for the multicast case. The complexity of the optimization for the multicast case is much lower than that for the general case. This is because in the optimization for the multicast case, variables $\mathbf{x}$ and $\boldsymbol{\beta}$ do not appear, and the constraints in (4), (5), (9), (10), (11) and (12) can be removed.

In the following, we show that a feasible linear network code can be obtained using a feasible solution to Problem 1 (e.g., using RLNC [3]), as illustrated in Section II-C.

Theorem 1: Suppose Problem 1 is feasible. Then, for each feasible $\mathbf{x}$ and $\boldsymbol{\beta}$, there exists a feasible linear network code design $\boldsymbol{\alpha}$ and $\mathbf{c}$ with a field size $F > T$ to deliver the desired flows to each terminal.

Proof: Please refer to Appendix A. ∎

Next, the minimum network cost of Problem 1 is no greater than the minimum costs of the two-step mixing approach for general connections in [27] and routing for integer flows, owing to the following reasons. Problem 1 with $\beta_{kij} = 1$ for all $(k, i), (i, j) \in \mathcal{E}$, instead of (5), is equivalent to the minimum-cost flow rate control problem in the second step of the two-step mixing approach for general connections in [27]. Problem 1 with an extra constraint $\sum_{p\in\mathcal{P}} x_{ij,p} \in \{0, 1\}$ for all $(i, j) \in \mathcal{E}$ is equivalent to the minimum-cost routing problem. Fig. 2 illustrates a feasible solution to Problem 1 that cannot be obtained by the two-step mixing approach [27] or routing. In this example, the minimum network cost of Problem 1 is smaller than those of the two-step mixing approach [27] and routing (which can be treated as infinity).

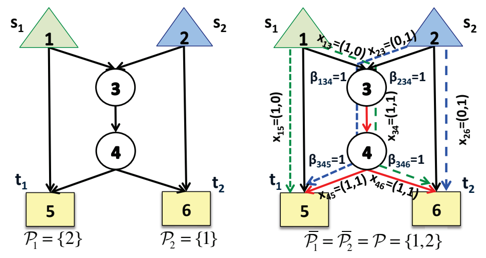


Fig. 3: Illustration of the network coding gain improvement of Problem 2 over Problem 1 for two unicasts over the butterfly network. Problem 1 (left) is not feasible, as the flows from s_1 to t_2 and s_2 to t_1 are not allowed to be mixed over edge $(3, 4)$ when $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$. However, Problem 2 (right) is feasible, as the flows are allowed to be mixed over edge $(3, 4)$ after the demand set expansion to $\mathcal{P}$.

When $\mathcal{P}_t \cap \mathcal{P}_{t'} = \emptyset$ for all $t \neq t'$ and $t, t' \in \mathcal{T}$ (e.g., multiple unicasts), Problem 1 for general connections reduces to the minimum-cost routing problem and cannot take advantage of the network coding gain. This is because using the network mixing representation, for decodability to hold, the extraneous flows of each terminal are not allowed to be mixed with the terminal's desired flows on the path to this terminal, thus limiting the network coding gain. To address this limitation, we now formulate Problem 2, which allows the expansion of the demand sets and the optimization over the expansions to increase the opportunity for mixing flows to different terminals. Let $\overline{\mathcal{P}_t}$ denote the expanded demand set, which satisfies $\mathcal{P}_t \subseteq \overline{\mathcal{P}_t} \subseteq \mathcal{P}$. Let $\overline{\boldsymbol{\mathcal{P}}} \triangleq (\overline{\mathcal{P}_t})_{t\in\mathcal{T}}$ denote the expanded demand sets of all the terminals.

Problem 2 (Mixing with Demand Set Expansion):

$$U^* = \min_{\{\overline{\mathcal{P}_t}\}} \quad U^*(\overline{\boldsymbol{\mathcal{P}}}) \qquad (15)$$

$$s.t. \quad \mathcal{P}_t \subseteq \overline{\mathcal{P}_t} \subseteq \mathcal{P}, \quad t \in \mathcal{T} \qquad (16)$$

where $U^*(\overline{\boldsymbol{\mathcal{P}}})$ is the optimal value to Problem 1 for $\overline{\boldsymbol{\mathcal{P}}}$.

The network coding gain improvement of Problem 2 can be easily understood from the case of two unicasts over the butterfly network, as illustrated in Fig. 3. By Theorem 1, we can easily show the following result.

Corollary 1: Suppose Problem 2 is feasible. Then, for each feasible solution, there exists a feasible linear network code design with a field size $F > T$ to deliver the desired flows to each terminal.

By comparing Problem 1 and Problem 2, we can obtain the following lemma.

Lemma 1: Suppose Problem 1 is feasible. Then, Problem 2 is feasible and $U^* \leq U^*(\boldsymbol{\mathcal{P}})$.

In the following sections, we focus on solving Problem 1 for given $\boldsymbol{\mathcal{P}}$. However, the obtained centralized and distributed algorithm can be easily extended to solve Problem 2 by further optimizing over $\overline{\boldsymbol{\mathcal{P}}}$. Later, in Section VI, we shall illustrate the results for Problem 1 and Problem 2 numerically.

IV. Centralized Algorithm

In this section, we develop a centralized algorithm to solve Problem 1, based on the concept of edge-disjoint flow paths discussed before. The centralized algorithm is conducted at a central node which is aware of global network information.

For all $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$, let N_p^t denote the number of flow paths from source s_p to terminal t. For Problem 1 to be feasible, assume $N_p^t > 0$. As illustrated in Section III, obtaining a feasible solution to Problem 1 is equivalent to selecting a set of flow paths satisfying certain requirements. Thus, we introduce flow path selection variables to indicate the flow paths selected for information transmission. Let $n_p^t \in \{1, \cdots, N_p^t\}$ denote the flow path selection variable for source s_p and terminal t (i.e., the index of the selected flow path from source s_p to terminal t). Denote the flow path selection variables as $\mathbf{n} \triangleq (n_p^t)_{p \in \mathcal{P}_t, t \in \mathcal{T}}$. In the following, we express variables $\mathbf{z}, \mathbf{f}, \mathbf{x}$ and $\boldsymbol{\beta}$ in terms of $\mathbf{n}$. To satisfy (6) and (8), we require n_p^t to take only one value from $\{1, \cdots, N_p^t\}$. Let $\mathcal{L}_p^t(n_p^t)$ denote the selected flow path (i.e., the set of edges on the selected flow path) from source s_p to terminal t. To satisfy (3) and (7) (or equivalently (14)), we require that the P_t flow paths from sources $\{s_p : p \in \mathcal{P}_t\}$ to terminal t are edge-disjoint, i.e.,

$$\mathcal{L}_p^t(n_p^t) \cap \mathcal{L}_{p'}^t(n_{p'}^t) = \emptyset, \ p, p' \in \mathcal{P}_t, \ p \neq p', \ t \in \mathcal{T}. \quad (17)$$

Then, variables $\mathbf{f}(\mathbf{n}) \triangleq \left(f_{ij,p}^t(n_p^t)\right)_{(i,j) \in \mathcal{E}, p \in \mathcal{P}_t, t \in \mathcal{T}}$ can be expressed in terms of variables $\mathbf{n}$ as follows:

$$f_{ij,p}^t(n_p^t) = \begin{cases} 1, & (i,j) \in \mathcal{L}_p^t(n_p^t) \\ 0, & \text{otherwise} \end{cases}, (i,j) \in \mathcal{E}, \ p \in \mathcal{P}_t, \ t \in \mathcal{T}. \quad (18)$$

By (7) and the monotonicity of $U_{ij}(\cdot)$, variables $\mathbf{z}(\mathbf{n}) \triangleq (z_{ij}(\mathbf{n}))_{(i,j) \in \mathcal{E}}$ can be chosen based on $\mathbf{f}(\mathbf{n})$ and expressed implicitly in terms of variables $\mathbf{n}$ as follows:

$$z_{ij}(\mathbf{n}) = \max_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}_t} f_{ij,p}^t(n_p^t), \ (i,j) \in \mathcal{E}. \quad (19)$$

In addition, to satisfy (4), (5), (9) and (11), $\boldsymbol{\beta}(\mathbf{n}) \triangleq \beta_{kij}(\mathbf{n})_{(k,i),(i,j) \in \mathcal{E}}$ can be chosen based on $\mathbf{f}(\mathbf{n})$ and expressed implicitly in terms of variables $\mathbf{n}$ as follows:

$$\beta_{kij}(\mathbf{n}) = \begin{cases} 1, & \max_{t \in \mathcal{T}, p \in \mathcal{P}_t} f_{ki,p}^t(n_p^t) f_{ij,p}^t(n_p^t) = 1 \\ 0, & \text{otherwise} \end{cases},$$
$$(k,i),(i,j) \in \mathcal{E}. \quad (20)$$

Algorithm 1 Centralized Algorithm for Problem 1

```
1: For all $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$, obtain all the flow paths $\{\mathcal{L}_p^t(n_p^t) : n_p^t \in \{1, \cdots, N_p^t\}\}$ from source $s_p$ to terminal $t$, using depth-first-search (DFS).
2: For all $t \in \mathcal{T}$, obtain the set of $P_t$ edge-disjoint flow paths $\mathcal{L}^t = \{(\mathcal{L}_p^t(n_p^t))_{p \in \mathcal{P}_t} : n_p^t \in \{1, \cdots, N_p^t\}$ for all $p \in \mathcal{P}_t$ and (17) is satisfied$\}$ from sources $\{s_p : p \in \mathcal{P}_t\}$ to terminal $t$.
3: Calculate the network costs of $L = \prod_{t \in \mathcal{T}} L^t$ combinations of $P_t$ edge-disjoint flow paths for all terminal $t \in \mathcal{T}$, and sort the $L$ combinations in the ascending order of their network costs, where $L^t = |\mathcal{L}^t|$ and the $l$-th combination is of the $l$-th smallest network cost $U_l$.
4: **initialize** $l = 1$ and $flag = 1$.
5: **while** $flag = 1$ **do**
6:     For all $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$, let $n_p^t$ denote the index of the flow path from source $s_p$ to terminal $t$ in the $l$-th combination.
7:     For all $p \in \mathcal{P}_t$, $t \in \mathcal{T}$ and $(i,j) \in \mathcal{E}$, set $f_{ij,p}^t(n_p^t)$ according to (18).
8:     For all $(i,j) \in \mathcal{E}$, set $z_{ij}(\mathbf{n})$ according to (19).
9:     For all $(k,i),(i,j) \in \mathcal{E}$, set $\beta_{kij}(\mathbf{n})$ according to (20).
10:    Based on $\{\beta_{kij}(\mathbf{n})\}$, (10) and (11), determine $\{x_{ij,p}(\mathbf{n})\}$ in the topological order.
11:    **if** (21) is satisfied **then**
12:        let $U_x^*(\boldsymbol{\mathcal{P}}) = U_l$, $\mathbf{z}^* = \mathbf{z}(\mathbf{n})$, $\mathbf{f}^* = \mathbf{f}(\mathbf{n})\}$, $\mathbf{x}^* = \mathbf{x}(\mathbf{n})\}$, $\boldsymbol{\beta}^* = \boldsymbol{\beta}(\mathbf{n})$, and set $flag = 0$
13:    **else**
14:        set $l = l + 1$
15:    **end if**
16: **end while**
```

Then, based on $\boldsymbol{\beta}(\mathbf{n})$, (10) and (11), $\mathbf{x}(\mathbf{n})$ can be determined in topological order[8] and expressed implicitly in terms of variables $\mathbf{n}$. Finally, to satisfy (12), we require

$$x_{it,p}(\mathbf{n}) = 0, \ i \in \mathcal{I}_t, \ p \notin \mathcal{P}_t, \ t \in \mathcal{T}. \quad (21)$$

Based on the above relationship between the flow path selection variables and the variables of Problem 1, we now describe the procedure of the centralized algorithm, i.e., Algorithm 1, which obtains the feasible flow paths of the minimum network cost.

Note that Constraints (6) and (8) are guaranteed in Step 1 and Step 8; Constraints (3) and (7) are guaranteed in Step 2 and Step 7; Constraints (4), (5), (9) are guaranteed in Step 9; Constraints (10) and (11) are guaranteed in Step 10; and Constraint (12) is considered in Steps 11–15. Therefore, we can see that the optimization to Problem 1 can be obtained by Algorithm 1.

V. Probabilistic Distributed Algorithms

In this section, using recent results for CSP [34], we develop two probabilistic distributed algorithms to solve Problem 1.

A. Background on Decentralized CSP

We review some existing results on CSP in [34].

Definition 3 (Constraint Satisfaction Problem): [34] A CSP consists of M variables $\{\lambda_1, \cdots, \lambda_M\}$ and K clauses

[8]A topological order of a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an ordering of its nodes such that for every directed edge $(i,j) \in \mathcal{E}$ from node $i \in \mathcal{V}$ to node $j \in \mathcal{V}$, i comes before j in the ordering. Such an order exists for the edges of any directed graph $\mathcal{G}$ that is acyclic.

$\{\phi_1, \cdots, \phi_K\}$. Each variable λ_m takes values in a finite set Λ, i.e., $\lambda_m \in \Lambda$ for all $m \in \mathcal{M} \triangleq \{1, \cdots, M\}$. Let $\boldsymbol{\lambda} \triangleq (\lambda_1, \cdots, \lambda_M) \in \Lambda^M$. Each clause $k \in \mathcal{K} \triangleq \{1, \cdots, K\}$ is a function $\phi_k : \Lambda^M \to \{0, 1\}$, where for an assignment of variables $\boldsymbol{\lambda} \in \Lambda^M$, $\phi_k(\boldsymbol{\lambda}) = 1$ if clause m is satisfied and $\phi_k(\boldsymbol{\lambda}) = 0$ otherwise. An assignment $\boldsymbol{\lambda} \in \Lambda^M$ is a solution to the CSP if and only if all clauses are simultaneously satisfied

$$\min_{k \in \mathcal{K}} \phi_k(\boldsymbol{\lambda}) = 1. \tag{22}$$

To solve a CSP in a distributed way, clause participation is introduced in [34]. Let $\boldsymbol{\lambda}_{-m} \triangleq (\lambda_1, \cdots, \lambda_{m-1}, \lambda_{m+1}, \cdots, \lambda_M) \in \Lambda^{M-1}$. For each variable λ_m, let $\mathcal{K}_m$ denote the set of clause indices in which it participates, i.e., $\mathcal{K}_m \triangleq \cup_{\boldsymbol{\lambda}_{-m} \in \Lambda^{M-1}} \{c : \min_{\lambda_m \in \Lambda} \phi_k(\lambda_m, \boldsymbol{\lambda}_{-m}) = 0, \max_{\lambda_m \in \Lambda} \phi_k(\lambda_m, \boldsymbol{\lambda}_{-m}) = 1\}$. Thus, we can rewrite the left hand side of (22) in a way that focuses on the satisfaction of each variable, i.e., $\min_{m \in \mathcal{M}} \min_{k \in \mathcal{K}_m} \phi_k(\boldsymbol{\lambda}) = 1$. This form enables us to solve CSPs in a distributed iterative way by locally evaluating the clauses in $\mathcal{K}_m$ and then updating λ_m.

CSPs are in general NP-complete and most effective CSP solvers are designed for centralized problems. The CFL algorithm [34, Algorithm 1], summarized in Algorithm 2, is a distributed iterative algorithm which can find a satisfying assignment to a CSP almost surely in finite time [34, Corollary 2]. Note that Algorithm 2 keeps a probability distribution over all possible values of each variable. The value of each variable is selected from this distribution. For each variable, if all the clauses in which a variable participates are satisfied with its current value, the associated probability distribution is updated to ensure that the variable value remains unchanged; if at least one clause is unsatisfied, the probability distribution evolves by interpolating between it and a distribution that is uniform on all values except the one that is currently generating dissatisfaction. Therefore, if all variables are simultaneously satisfied in all clauses, the same assignment of values will be reselected indefinitely with probability 1.

Algorithm 2 Communication-Free Learning [34]

```
1: Initialize $q_m(\lambda) = \frac{1}{|\Lambda|}$ for all $\lambda \in \Lambda$, where $|\Lambda|$ denotes the cardinality of $\Lambda$.
2: loop
3:     Realize a random variable, selecting $\lambda_m = \lambda$ with probability $q_m(\lambda)$.
4:     Evaluate $\min_{k \in \mathcal{K}_m} \phi_k(\boldsymbol{\lambda})$, returning satisfied if its value is 1 and unsatisfied otherwise.
5:     if satisfied then
6:         set $q_m(\lambda) = \begin{cases} 1, & \text{if } \lambda = \lambda_m \\ 0, & \text{otherwise} \end{cases}$
7:     else
8:         set $q_m(\lambda) = \begin{cases} (1-b)q_m(\lambda) + \frac{a}{|\Lambda|-1+a/b}, & \text{if } \lambda = \lambda_m \\ (1-b)q_m(\lambda) + \frac{b}{|\Lambda|-1+a/b}, & \text{otherwise} \end{cases}$,
           where $a, b \in (0, 1]$ are design parameters.
9:     end if
10: end loop
```

B. Path-based Probabilistic Distributed Algorithm

In this part, we develop a path-based probabilistic distributed algorithm to solve Problem 1, using recent results in [34]. This distributed algorithm is based on the concept of edge-disjoint flow paths discussed before. It can be viewed as a distributed version of the path-based centralized algorithm, i.e., Algorithm 1. For all $p \in \mathcal{P}_t$, $t \in \mathcal{T}$ and $n_p^t \in \{1, \cdots, N_p^t\}$, this algorithm requires each node on the n_p^t-th flow path from source s_p to terminal t to know its neighboring edge on this flow path. Note that it is not necessary for each node on the n_p^t-th flow path to be aware of other edges on the n_p^t-th flow path.

First, we construct a path-based CSP corresponding to the feasibility problem obtained from Problem 1. Treat $\mathbf{n}$ as the variables of the path-based CSP, where $n_p^t \in \{1, \cdots, N_p^t\}$ denotes the index of the selected flow path from source s_p to terminal t. As illustrated in Section IV, the constraints of $\mathbf{f}$ in (6) and (8) for Problem 1 can be taken into account by choosing $n_p^t \in \{1, \cdots, N_p^t\}$, for all $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$. The constraints in (3) and (7) (or equivalently (14)) can be replaced by the constraint of $\mathbf{n}$ in (17) for the path-based CSP. Variables $\{\beta_{ijk}(\mathbf{n})\}$ and $\{x_{ij,p}(\mathbf{n})\}$ can be determined for given $\mathbf{n}$ via (18), (20), (10) and (11). Thus, the last constraint in (12) of Problem 1 can be replaced by the constraint of $\mathbf{n}$ for the path-based CSP in (21). Therefore, we can write the clause for n_p^t as follows:

$$\phi_p^{n,t}(\mathbf{n}) = \begin{cases} 1, & \text{if (17) and (21) hold} \\ 0, & \text{otherwise} \end{cases}, \ p \in \mathcal{P}_t, \ t \in \mathcal{T}. \tag{23}$$

We thus have the following proposition.[9]

Proposition 1 (Path-based CSP): The path-based CSP with variables $\mathbf{n}$ ($n_p^t \in \{1, \cdots, N_p^t\}$) and clauses (23) has considered all the constraints in Problem 1.

Now, we present a path-based distributed probabilistic algorithm, i.e., Algorithm 3, to obtain a feasible solution to the path-based CSP using CFL [34, Algorithm 1]. Based on the convergence result of CFL [34, Corollary 2], we know that Algorithm 3 can find a feasible solution to Problem 1 in almost surely finite time. Fig. 4 illustrates the convergence of Algorithm 3. From Fig. 4, we can see that Algorithm 3 converges to a feasible solution (i.e., the feasible solution illustrated in Fig. 2 (a)) to Problem 1 for the network in Fig. 2 quite quickly (within 35 iterations). This feasible solution corresponds to flow paths $1-3-8$, $1-3-4-6-7$, $2-5-7$, $1-3-9-10$ and $2-5-4-6-10$. The network cost of this feasible solution is 11.

Relying on Algorithm 3, we present a path-based distributed probabilistic algorithm, Algorithm 4, to obtain the optimal solution to Problem 1 among multiple feasible solutions obtained by Algorithm 3.[10] Since Algorithm 3 can find any feasible solution

[9]Note that the clauses of the path-based CSP cannot be further partitioned, as all the variables $\mathbf{n}$ are coupled in general.

[10]In Step 3 of Algorithm 4, the path-based CFL is run for a sufficiently long time. Step 6 of Algorithm 4 can be implemented with a master node obtaining the network cost of the path-based CFL from all nodes or with all nodes computing the average network cost of the path-based CFL locally via a gossip algorithm.

Algorithm 3 Path-based CFL

```
1:  For all $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$, obtain all the flow paths from source $s_p$ to terminal $t$, using DFS.
2:  For all $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$, source $s_p$ initializes $q_p^t(n) = \frac{1}{N_p^t}$ for all $n \in \{1, ..., N_p^t\}$.
3:  loop
4:      For all $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$, source $s_p$ realizes a random variable, selecting $n_p^t = n$ with probability $q_p^t(n)$, where $n \in \{1, ..., N_p^t\}$, and sends signaling packet $(p, t, n_p^t)$ over edge $(s_p, j) \in \mathcal{L}_p^t(n_p^t)$ to node $j$. Once node $j$ receives signaling packet $(p, t, n_p^t)$ over edge $(i, j) \in \mathcal{L}_p^t(n_p^t)$ , it forwards this signaling packet to node $k$ over edge $(j, k) \in \mathcal{L}_p^t(n_p^t)$.
5:      For all $(i, j) \in \mathcal{E}$, if edge $(i, j)$ receives signaling packets $\{(p, t, n_p^t) : p \in \mathcal{P}_t\}$ to terminal $t \in \mathcal{T}$ from more than one source in $\{s_p : p \in \mathcal{P}_t\}$, it sends NAK back to each of these sources along its selected path $n_p^t$.
6:      For all $p \in \mathcal{P}_t$, $t \in \mathcal{T}$ and $(i, j) \in \mathcal{E}$, set $f_{ij,p}^t(n_p^t)$ according to (18).
7:      For all $(k, i), (i, j) \in \mathcal{E}$, set $\beta_{kij}(\mathbf{n})$ according to (20).
8:      Based on $\boldsymbol{\beta}(\mathbf{n})$, (10) and (11), determine $\mathbf{x}(\mathbf{n})$ in topological order.
9:      Every terminal $t \in \mathcal{T}$ checks (21). For all $i \in \mathcal{I}_t$ and $p \notin \mathcal{P}_t$, if (21) is unsatisfied, terminal $t$ sends NAK back to source $s_p$ (along any path to source $s_p$), and sends NAK back to each of the sources in $\mathcal{P}_t$ (along its selected path).
10:     for $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$ do
11:         if source $s_p$ receives no NAKs then
12:             set $q_p^t(n) = \begin{cases} 1, \text{if } n = n_p^t \\ 0, \text{otherwise} \end{cases}$
13:         else
14:             set $q_p^t(n) = \begin{cases} (1-b)q_p^t(n) + \frac{a}{N_p^t - 1 + \frac{a}{b}}, \text{if } n = n_p^t \\ (1-b)q_p^t(n) + \frac{b}{N_p^t - 1 + \frac{a}{b}}, \text{otherwise} \end{cases}$,
                where $a, b \in (0, 1]$ are design parameters.
15:         end if
16:     end for
17: end loop
```

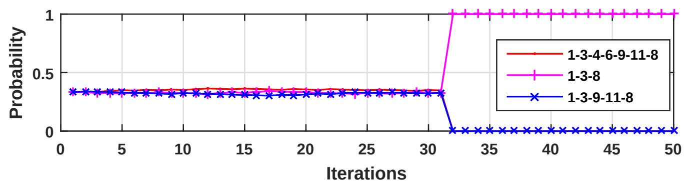
Algorithm 4 Path-based Distributed Algorithm

```
1:  $l$ = 1 and $U_1$ = $+\infty$.
2:  loop
3:      Run the path-based CFL in Algorithm 3 to the path-based CSP corresponding to Problem 1. Let $\mathbf{n}_l$ denote the feasible solution obtained by Algorithm 3 and let $\bar{U}_l$ denote the corresponding network cost.
4:      if $\bar{U}_l < U_l$ then
5:          set $U_{l+1} = \bar{U}_l$, $\mathbf{n}^* = \mathbf{n}_l$, and $l = l + 1$.
6:      end if
7:  end loop
```

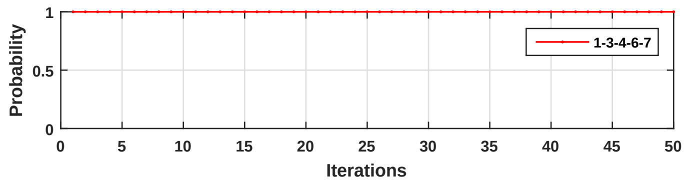
to Problem 1 with positive probability, $U_l \to U^*(\{\mathcal{P}_t\})$ almost surely as $l \to \infty$, where U_l denotes the minimum network cost obtained by the first l path-based CFLs. Fig. 5 illustrates the convergence of Algorithm 4. From Fig. 5, we can see that Algorithm 4 obtains the optimal network cost 11 to Problem 1 for the network in Fig. 2 quite quickly (within 5 iterations).

C. Edge-based Probabilistic Distributed Algorithm

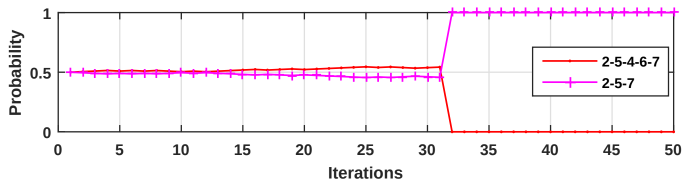
In this part, we develop an edge-based probabilistic distributed algorithm to solve Problem 1, using recent results in [34]. Compared with the path-based distributed algorithm in Section V-C, this edge-based distributed algorithm does not



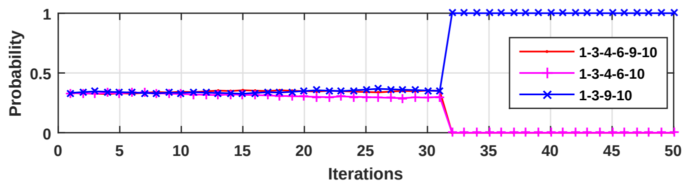
(a) Flow path from source s_1 to terminal t_1.



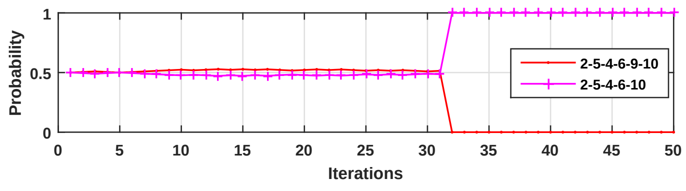
(b) Flow path from source s_1 to terminal t_2.



(c) Flow path from source s_2 to terminal t_2.



(d) Flow path from source s_1 to terminal t_3.



(e) Flow path from source s_2 to terminal t_3.

Fig. 4: Convergence of the path-based CFL in Algorithm 3 for Problem 1 of the network in Fig. 2. a = 1 and b = 0.01. These convergence curves are for one realization of the random Algorithm 3. Note that all the flow paths are shown in the figure.

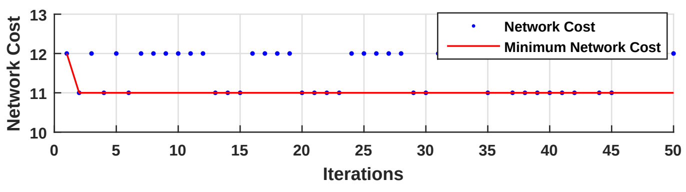


Fig. 5: Network costs of the path-based CFLs in Algorithm 4 for Problem 1 of the network in Fig. 2. Each blue dot represents the network cost of a feasible solution obtained by the path-based CFL in each iteration of Algorithm 4. While the red curve represents the minimum network cost obtained by Algorithm 4 within a certain number of iterations. The blue dots and red curve are for one realization of the random Algorithm 4.

require any path information.

Obtaining a feasible solution to Problem 1 can be directly treated as a CSP [34]. Specifically, $\mathbf{z}, \mathbf{f}, \mathbf{x}, \boldsymbol{\beta}$ and $\{0,1\}$ can be treated as the variables and the finite set of the CSP. Constraints (7)-(12) can be treated as the clauses of the CSP. While CSPs are in general NP-complete, several centralized CSP solvers (see references in [34]) and the distributed CSP solver proposed in [34] can be applied to solve this (naïve) CSP. However, the direct application of the distributed CSP solver in [34] leads to high complexity owing to the large constraint set. In this part, by exploring the features of the constraints in Problem 1, we obtain a different CSP and present a probabilistic distributed solution with a significantly reduced number of clauses.

First, we construct a new problem, which we show to be a CSP. This new problem is better suited than the original problem to being treated using a probabilistic distributed algorithm based on the distributed CSP solver presented in [34]. Combining (3) and (7), we have an equivalent constraint purely in terms of $\mathbf{f}$, i.e., (14). In addition, from (11), we have an equivalent constraint purely in terms of $\mathbf{x}$, i.e.,

$$\exists\ \beta_{kij} \in \{0,1\}\ \forall k \in \mathcal{I}_i, \text{ s.t. } \mathbf{x}_{ij} = \vee_{k\in\mathcal{I}_i}\beta_{kij}\mathbf{x}_{ki}, \quad (i,j)\in\mathcal{E},\ i\notin\mathcal{S}. \tag{24}$$

Therefore, we can solve only for variables $\mathbf{f}$ and $\mathbf{x}$ in a distributed way, as $\mathbf{z}$ can be obtained directly from feasible $\mathbf{f}$ by choosing $z_{ij} = \max_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}_t} f^t_{ij,p}$ according to (3) and (7), and $\boldsymbol{\beta}$ can be obtained from feasible $\mathbf{x}$ by (10) and (11). We group all the local variables for each edge $(i,j)\in\mathcal{E}$ and introduce the vector variable $(\mathbf{f}_{ij},\mathbf{x}_{ij}) \in \mathcal{Y}_{ij}$, where $\mathbf{f}_{ij} \triangleq \left(\mathbf{f}^t_{ij}\right)_{t\in\mathcal{T}}$, $\mathbf{f}^t_{ij} \triangleq \left(f^t_{ij,p}\right)_{p\in\mathcal{P}_t}$ and $\mathcal{Y}_{ij} \triangleq \{(\mathbf{f}_{ij},\mathbf{x}_{ij}) : (4),(6),(9),(10),(12),(14)\}$. We also write $\mathcal{Y}_{ij} = \{\mathbf{y}_{ij,1},\cdots,\mathbf{y}_{ij,Y_{ij}}\}$, where $Y_{ij} = |\mathcal{Y}_{ij}|$. We now consider a new CSP, different from the naïve one that would be directly obtained from Problem 1. We treat $(\mathbf{f}_{ij},\mathbf{x}_{ij})$ and $\mathcal{Y}_{ij}$ as the variable and the finite set for edge (i,j) of the CSP. We write the clauses for $\{(\mathbf{f}_{ij},\mathbf{x}_{ij})\}$ as follows:

$$\phi^f_i(\mathbf{f}_i) = \begin{cases} 1, & \text{if (8) holds } \forall p\in\mathcal{P}_t,\ t\in\mathcal{T} \\ 0, & \text{otherwise}\end{cases},\ i\in\mathcal{V} \tag{25}$$

$$\phi^x_{ij}(\mathbf{x}_{ij},\{\mathbf{x}_{ki}:k\in\mathcal{I}_i\}) = \begin{cases} 1, & \text{if (24) holds} \\ 0, & \text{otherwise}\end{cases}, \quad (i,j)\in\mathcal{E}, i\notin\mathcal{S} \tag{26}$$

where $\mathbf{f}_i \triangleq (\mathbf{f}_{ik})_{k\in\mathcal{O}_i,k\in\mathcal{I}_i}$. Note that the local constraints in (4), (6), (9), (10), (12) and (14) (i.e., (3) and (7)) are considered in the finite set $\mathcal{Y}_{ij}$ of the CSP with respect to each edge $(i,j)\in\mathcal{E}$. On the other hand, the non-local constraints in (8) and (24) are considered in clauses ϕ^f_i in (25) and ϕ^x_{ij} in (26), respectively. We thus have the following proposition.

Proposition 2 (Edge-based CSP): The edge-based CSP with variables $(\mathbf{f}_{ij},\mathbf{x}_{ij})\in\mathcal{Y}_{ij}$, $(i,j)\in\mathcal{E}$ and clauses (25) and (26) has considered all the constraints in Problem 1.

Note that the number of variables (E) and the number of clauses ($\leq V+E-P$) of the new CSP are much smaller than the number of variables ($\leq (1+D+P+TP)E$) and the number of clauses ($\leq (1+T+TP)E+TPV+TPD$) of the naïve CSP directly obtained from Problem 1. This feature will favor the complexity reduction of a distributed solution based on the distributed CSP solver in [34].

Algorithm 5 Edge-based CFL

```
1: For all $(i,j)\in\mathcal{E}$, edge $(i,j)$ initializes $q_{ij}(\mathbf{y}) = \frac{1}{Y_{ij}}$ for all $\mathbf{y}\in\mathcal{Y}_{ij}$.
2: loop
3:    For all $(i,j)\in\mathcal{E}$, edge $(i,j)$ realizes a random variable, selecting $(\mathbf{f}_{ij},\mathbf{x}_{ij}) = \mathbf{y}$ with probability $q_{ij}(\mathbf{y})$, where $\mathbf{y}\in\mathcal{Y}_{ij}$.
4:    for $(i,j)\in\mathcal{E}$ do
5:       Each edge $(i,j)$ evaluates all the clauses in $\Phi_{ij}$.
6:       if all clauses in $\Phi_{ij}$ are satisfied then
7:          set $q_{ij}(\mathbf{y}) = \begin{cases}1, & \text{if } \mathbf{y}=(\mathbf{f}_{ij},\mathbf{x}_{ij})\\ 0, & \text{otherwise}\end{cases}$
8:       else
9:          set $q_{ij}(\mathbf{y}) = \begin{cases}(1-b)q_{ij}(\mathbf{y}) + \frac{a}{Y_{ij}-1+a/b}, & \mathbf{y}=(\mathbf{f}_{ij},\mathbf{x}_{ij})\\ (1-b)q_{ij}(\mathbf{y}) + \frac{b}{Y_{ij}-1+a/b}, & \text{otherwise}\end{cases}$,
            where $a,b\in(0,1]$ are design parameters.
10:      end if
11:   end for
12: end loop
```

Next, we construct the clause partition. The set of clauses in which variable $(\mathbf{f}_{ij},\mathbf{x}_{ij})$ participates is

$$\Phi_{ij} = \left\{\phi^f_i,\phi^f_j\right\}\cup\left\{\phi^x_{ij},\phi^x_{jk} : i\notin\mathcal{S}, k\in\mathcal{O}_j\right\},\ (i,j)\in\mathcal{E}. \tag{27}$$

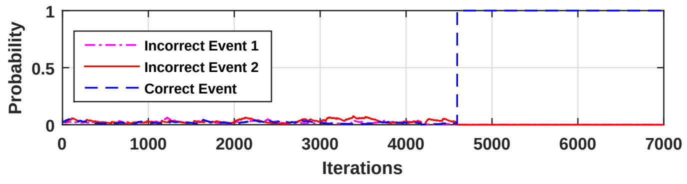
Then, the focus can be on the satisfaction of each variable $(\mathbf{f}_{ij},\mathbf{x}_{ij})$, i.e., the satisfaction of each set of clauses Φ_{ij}. Now, the new CSP can be solved using the distributed iterative CFL algorithm [34, Algorithm 1]. Specifically, each edge $(i,j)\in\mathcal{E}$ realizes a random variable selecting $(\mathbf{f}_{ij},\mathbf{x}_{ij})$. Allow message passing on $(\mathbf{f}_{ij},\mathbf{x}_{ij})$ between adjacent nodes to evaluate the related clauses. Based on whether the clauses in (27) are satisfied or not, the distribution of the random variable of each edge $(i,j)\in\mathcal{E}$ is updated. The details are summarized in Algorithm 5, which obtains a feasible solution to the edge-based CSP using CFL [34, Algorithm 1]. Based on the convergence result of CFL [34, Corollary 2], we know that Algorithm 5 can find a feasible solution to Problem 1 in almost surely finite time. Fig. 6 illustrates the convergence of Algorithm 5. From Fig. 6, we can see that Algorithm 5 converges to a feasible solution to Problem 1 for the network in Fig. 2 within 5000 iterations. This feasible solution is the same as the one shown in Fig. 4, with network cost 11.

Relying on Algorithm 5, we present an edge-based distributed probabilistic algorithm, Algorithm 6, to solve Problem 1.[11] Since Algorithm 5 can find any feasible solution to Problem 1 with positive probability, $U_l \to U^*(\{\mathcal{P}_t\})$ almost surely as $l\to\infty$, where U_l denotes the smallest network cost obtained by the first l edge-based CFLs. Fig. 7 illustrates the convergence of Algorithm 6. From Fig. 7, we can see that Algorithm 6 obtains the optimal network cost 11 to Problem 1 for the network in Fig. 2 quite quickly (within 5 iterations).

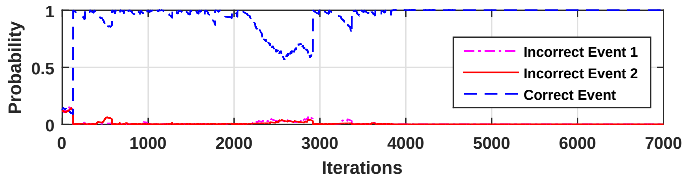
[11]Note that Step 3 and Step 6 of Algorithm 6 can be implemented in similar ways to those in Algorithm 4.

Algorithm 6 Edge-based Distributed Algorithm

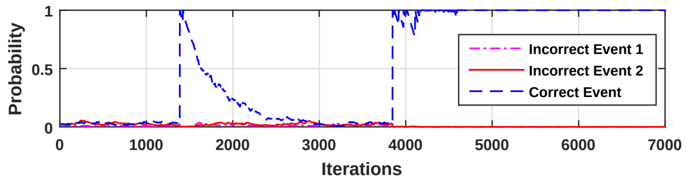
```
1: $l = 1$ and $U_1 = +\infty$.
2: loop
3:     Run the edge-based CFL in Algorithm 5 to the edge-based CSP corresponding to Problem 1. Let $\{(\mathbf{f}_{ij,l}, \mathbf{x}_{ij,l}) : (i,j) \in \mathcal{E}\}$ denote the feasible solution obtained by Algorithm 5 and let $\bar{U}_l$ denote the corresponding network cost.
4:     if $\bar{U}_l < U_l$ then
5:         set $U_{l+1} = \bar{U}_l$, $(\mathbf{f}^*_{ij}, \mathbf{x}^*_{ij}) = (\mathbf{f}_{ij,l}, \mathbf{x}_{ij,l})$ for all $(i,j) \in \mathcal{E}$, and $l = l+1$.
6:     end if
7: end loop
```



(a) Variable $(\mathbf{f}_{25}, \mathbf{x}_{25})$ for edge $(2,5)$.



(b) Variable $(\mathbf{f}_{57}, \mathbf{x}_{57})$ for edge $(5,7)$.



(c) Variable $(\mathbf{f}_{67}, \mathbf{x}_{67})$ for edge $(6,7)$.

Fig. 6: Convergence of the edge-based CFL in Algorithm 5 for Problem 1 of the network in Fig. 2. $a = 1$ and $b = 0.01$. Note that for each edge, the "Correct Event" indicates the variable taking the value which corresponds to the feasible solution obtained by the edge-based CFL. These convergence curves are for one realization of the random Algorithm 5.

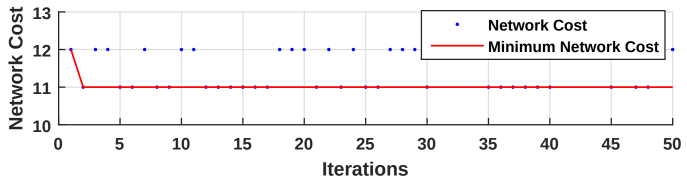


Fig. 7: Network costs of the edge-based CFLs in Algorithm 4 for Problem 1 of the network in Fig. 8. Each blue dot represents the network cost of a feasible solution obtained by the edge-based CFL in each iteration of Algorithm 4. While the red curve represents the minimum network cost obtained by Algorithm 4 within a certain number of iterations. The blue dots and red curve are for one realization of the random Algorithm 4.

D. Comparison

In this part, we compare the path-based and edge-based distributed algorithms. In obtaining a feasible solution to Problem 1, the path-based CFL, i.e., Algorithm 3 and the edge-based CFL, i.e., Algorithm 5 both base on CFL [34, Algorithm 1]. The convergence result of CFL [34, Corollary 2] guarantee that Algorithm 3 and Algorithm 5 both converge to feasible solutions to Problem 1 almost surely in finite time. However, Algorithm 3 converges much faster than Algorithm 5 in our simulations. This is expected, as Algorithm 3 solves a path-based CSP, while Algorithm 5 solves an edge-based CSP. The number of variables and the number of possible values for each variable for the path-based CSP are much smaller than those for the edge-based CSP. The difference in the convergence rates of Algorithm 3 and Algorithm 5 can be seen by comparing Fig. 4 and Fig. 6. On the other hand, Algorithm 3 requires more local information than Algorithm 5. In particular, Algorithm 3 requires all the nodes on one path from a source node to a terminal node to be aware of their neighboring nodes on the path (not all the nodes on the path). Algorithm 5 instead only requires each node to be aware of its neighboring nodes.

In obtaining an optimal solution to Problem 1 among multiple feasible solutions, the path-based distributed algorithm, i.e., Algorithm 4 and the edge-based distributed algorithm, i.e., Algorithm 6 base on the path-based CFL, i.e., Algorithm 3 and the edge-based CFL, i.e., Algorithm 5, respectively, in the same way. Therefore, Algorithm 4 and Algorithm 6 share similar convergence properties. This can be illustrated in Fig. 5 and Fig. 7.

VI. Numerical Illustration

In this section, we numerically illustrate the performance of the proposed optimal solutions to Problems 1 and 2 using mixing only with the two-step mixing approach in [27] and optimal routing for general connections of integer flows.

In the simulation, we consider the Sprint backbone network [35] as illustrated in Fig. 8. We choose sources $\mathcal{S} = \{8, 11\}$ and terminals $\mathcal{T} \subseteq \{2, 3, 4, 6, 9\}$. The edge directions are chosen to permit connections and help illustrate network coding gain. The green edges have edge cost 1, while the blue edges have edge cost 10 or 20. The edge costs are chosen to make the network coding advantage exist at least for some connection requests [31]. Note that network coding gain takes effect only if transmitting coded information requires a lower network cost than routing. We consider 1000 random realizations of demand sets. For each realization, a pair or triplet of terminals (i.e., $T = 2, 3$) are selected from $\{2, 3, 4, 6, 9\}$ uniformly at random, and each selected terminal randomly, uniformly, independently demands a source out of the two sources in $\mathcal{S} = \{8, 11\}$. In addition, each selected terminal randomly chooses to demand the other source or not according to a Bernoulli distribution with probability $q - 1$ of selecting a second source, where $q \in [1, 2]$. Thus, q represents the expected number of sources selected by each terminal (i.e., P_t). Note that $q = 2$ indicates multicast, and $q = 1$ results in unicast connections. In this way, general connections are randomly generated with q controlling the average size of the intersections of the demand sets by different terminals.

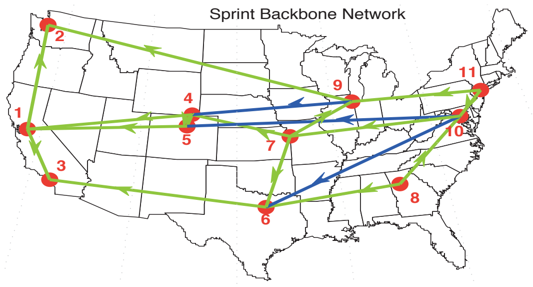


Fig. 8: Sprint backbone network topology [35]. $\mathcal{S} = \{8, 11\}$ and $\mathcal{T} \subseteq \{2, 3, 4, 6, 9\}$. The edge costs are: 20 for edges $(10, 5)$ and $(10, 6)$, 10 for edge $(9, 4)$, and 1 for all the other edges.

	T=2		T=3	
	q=1.2	q=1.8	q=1.2	q=1.8
Problem 2	7.49	12.70	12.82	20.79
Problem 1	8.99	13.80	16.96	23.20
Two-step Mixing [27]	8.99	13.80	17.24	24.94
Routing	9.36	18.68	17.25	32.34

TABLE I: Average optimal network cost of the network in Fig. 8.

A. Network Cost

Table. I illustrates the average optimal network cost (averaged over 1000 random realizations) for different q and T. Note that the optimal network costs of Problems 1 and 2 are obtained by the centralized algorithm, i.e., Algorithm 1. We can observe that the average optimal network costs of all the schemes increase with increases of q or T, i.e., the increase of network load. The average network costs of the optimal solutions to Problems 1 and 2 are lower than the optimal routing, with average cost reductions up to 28% and 36%, respectively. The average cost reductions are due to the network coding gain exploited by Problems 1 and 2. Specifically, edge $(10, 7)$ can serve as the coding edge for the butterfly subnetwork consisting of nodes 6, 7, 8, 9, 10 and 11, and edges $(7, 4)$ and $(4, 1)$ can serve as the coding edge for the butterfly subnetwork consisting of nodes 1, 2, 3, 4, 6, 7 and 9, in the Sprint backbone network in Fig. 8. The network coding gain increases as q or T increases. This is because, using network coding, edges can be used more efficiently in the case of high network load.

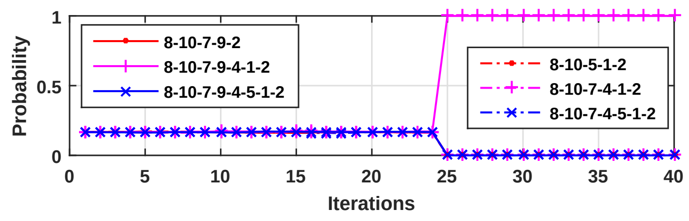
In addition, the average network costs of the optimal solutions to Problems 1 and 2 are lower than the two-step mixing approach, with average cost reductions up to 7% and 26%, respectively. The average cost reductions are due to the extra network coding gain (achieved through mixing) exploited by Problems 1 and 2. Specifically, given the demand sets of all the terminals, mixing or not in the two-step mixing approach (determined in the first step, separately from the second flow rate control step) is restricted by all the physical paths, while mixing or not in Problems 1 and 2 (determined jointly with flow rate control) is only restricted by the actual paths that each flow will take, which is also illustrated in the example in Fig. 2. Note that the average network cost of Problem 1 is lower than the two-step mixing when $T = 3$. The average cost reductions of the optimal solutions to Problems 1 and 2 increase as T increases, as there are more physical paths to terminals restricting network coding (mixing) in the two-step mixing approach.

On the other hand, the average network cost of the optimal solution to Problem 2 is lower than that of the optimal solution to Problem 1, with average cost reduction up to 24%, illustrating the consequence of Lemma 1. For a given T, the performance gain of Problem 2 over Problem 1 decreases as q increases, since the difference between the feasibility regions of the two problems reduces with the increase of q. Note that when $q = 2$ (i.e., multicast), the two problems (feasibility regions) are the same. However, for a given q, the performance gain of Problem 2 over Problem 1 increases as T increases, since the difference between the feasibility regions of the two problems increases with the increase of T.

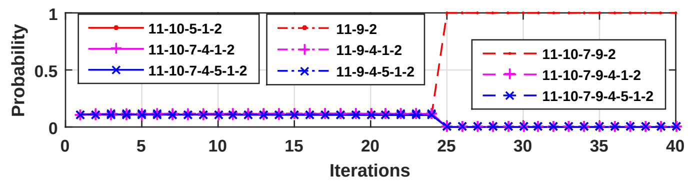
B. Convergence

We illustrate the convergence performance of own distributed Algorithm 4. Consider $s_1 = 8$, $s_2 = 11$, $t_1 = 2$, $t_2 = 6$, $\mathcal{P}_1 = \{1, 2\}$, $\mathcal{P}_2 = \{2\}$ and $\mathcal{P} = \{1, 2\}$. In this case, the optimal network costs of Problem 2, Problem 1, the two-step mixing approach and routing are 10, 28, 28, 28, respectively. The optimal network mixing solution to Problem 2 is achieved through the demand set expansion, i.e., $\bar{\mathcal{P}}_1 = \bar{\mathcal{P}}_2 = \mathcal{P} = \{1, 2\}$. The expanded demand set corresponds to multicast, where the network coding gain is achieved. The optimal mixing (coding) solutions with cost 10 corresponds to flow paths $8-10-7-4-1-2$, $11-10-7-9-2$ $(11-9-2)$, $8-6$ and $11-10-7-6$. There is no network mixing (coding) solution to Problem 1 and the two-step mixing approach. There are three optimal routing solutions of cost 28, which are also optimal (feasible but non-coding) solutions for Problem 1 and the two-step mixing approach. The first one corresponds to flow paths $8-10-5-1-2$, $11-10-7-9-2$ and $11-10-7-6$. The second one corresponds to flow paths $8-10-7-4-1-2$, $11-9-2$ and $11-10-6$. The third one corresponds to flow paths $8-10-5-1-2$, $11-9-2$ and $11-10-7-6$. In the following, we illustrate the convergence for the path-based and edge-based distributed algorithms for Problem 2 (at $\bar{\mathcal{P}}_1 = \bar{\mathcal{P}}_2 = \mathcal{P} = \{1, 2\}$), respectively.

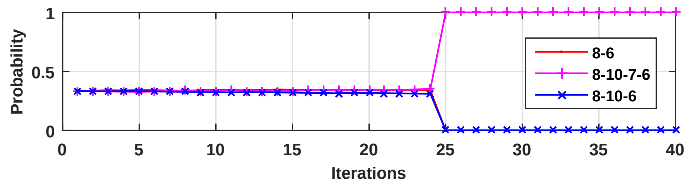
1) Path-based Probabilistic Distributed Algorithm: Fig. 9 illustrates the convergence of Algorithm 3 (i.e., Step 3 in Algorithm 4). From Fig. 9, we can see that Algorithm 3 converges to a feasible solution to Problem 2 quite quickly (within 25 iterations). This feasible solution corresponds to flow paths $8-10-7-4-1-2$, $11-10-7-9-2$, $8-10-7-6$ and $11-10-6$. The network cost of this feasible solution is 10, i.e., the optimal network cost to Problem 2. Fig. 10 illustrates the convergence of Algorithm 4 for one instance. We can see that there exist multiple feasible mixing solutions to Problem 2, which are of different network costs, and running Algorithm 3 for multiple times can result in different feasible solutions. Thus, the minimum network cost may decrease as the number of iterations increases. Algorithm 4 obtains the optimal network cost 10 to Problem 2 quite quickly (within 100 iterations). Fig. 11 illustrates the average convergence of Algorithm 4 over 1000 instances. We can see that on average, within 100 iterations, the minimum network cost under



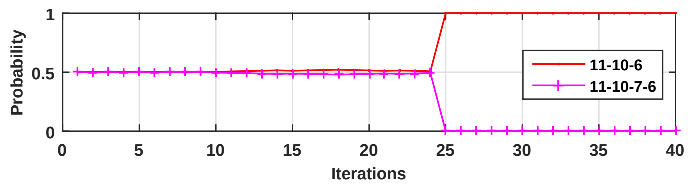
(a) Flow path from (source) node 8 to (terminal) node 2.



(b) Flow path from (source) node 11 to (terminal) node 2.



(c) Flow path from (source) node 8 to (terminal) node 6.



(d) Flow path from (source) node 11 to (terminal) node 6.

Fig. 9: Convergence of the path-based CFL in Algorithm 3 for Problem 2 of the network in Fig. 8. $a = 0.05$ and $b = 0.009$. These convergence curves are for one realization of the random Algorithm 3. Note that all the flow paths are shown in the figure.

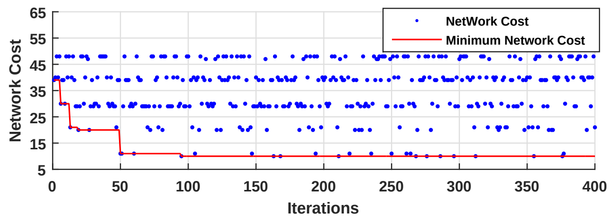


Fig. 10: Network costs of Algorithm 4 for Problem 2 of the network in Fig. 8. Each blue dot represents the network cost of a feasible solution obtained by the path-based CFL in each iteration of Algorithm 4. While the red curve represents the minimum network cost obtained by Algorithm 4 within a certain number of iterations. The blue dots and red curve are for one realization of the random Algorithm 4.

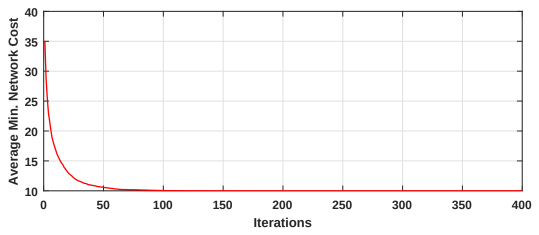
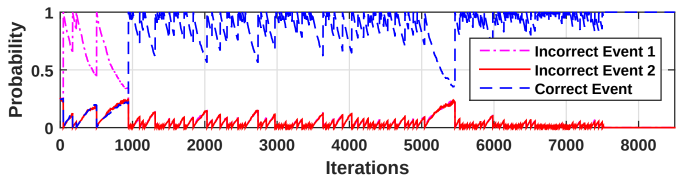
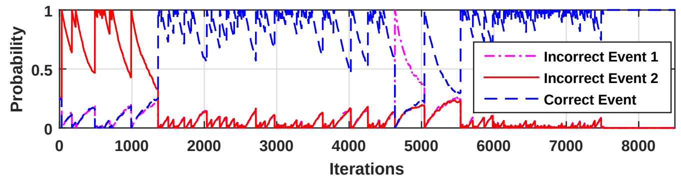


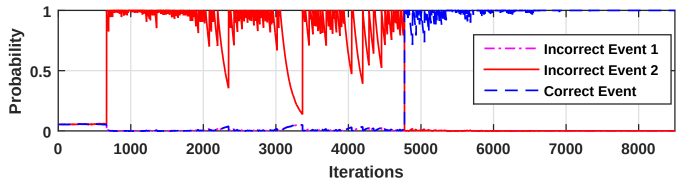
Fig. 11: Average minimum network costs of the path-based CFLs in Algorithm 4 for Problem 2 of the network in Fig. 8 over 1000 instances. The red curve here represents the average of the red curves in Fig. 10 over 1000 instances.



(a) Variable $(\mathbf{f}_{810}, \mathbf{x}_{810})$ for edge $(8, 10)$.



(b) Variable $(\mathbf{f}_{1110}, \mathbf{x}_{1110})$ for edge $(11, 10)$.



(c) Variable $(\mathbf{f}_{12}, \mathbf{x}_{12})$ for edge $(1, 2)$.

Fig. 12: Convergence of the edge-based CFL in Algorithm 5 for Problem 2 of the network in Fig. 8. $a = 1$ and $b = 0.01$. Note that for each edge, the "Correct Event" indicates the variable taking the value which corresponds to the feasible solution obtained by the edge-based CFL. These convergence curves are for one realization of the random Algorithm 5.

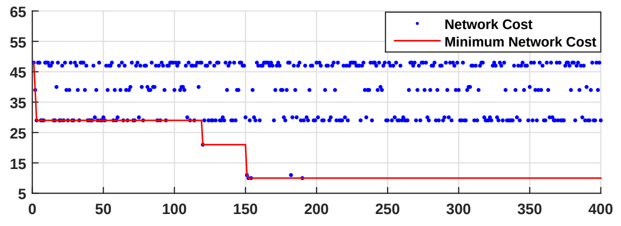


Fig. 13: Network costs of Algorithm 6 for Problem 2 of the network in Fig. 8. Each blue dot represents the network cost of a feasible solution obtained by the edge-based CFL in each iteration of Algorithm 6. While the red curve represents the minimum network cost obtained by Algorithm 6 within a certain number of iterations. The blue dots and red curve are for one realization of the random Algorithm 6.

Algorithm 4 converges to 10, which is the optimal network cost to Problem 2 obtained by the centralized algorithm in Algorithm 1.

2) Edge-based Probabilistic Distributed Algorithm: Fig. 12 illustrates the convergence of Algorithm 5 (i.e., Step 3 in Algorithm 6). From Fig. 12, we can see that Algorithm 5 converges to a feasible solution to Problem 2 within 8000

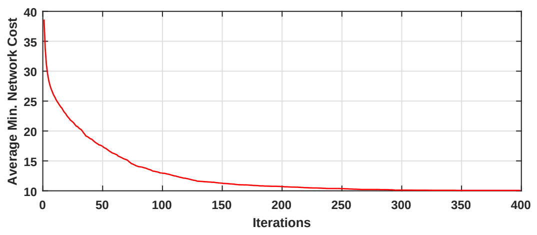


Fig. 14: Average minimum network costs of the edge-based CFLs in Algorithm 6 for Problem 2 of the network in Fig. 8 over 1000 instances. The red curve here represents the average of the red curves in Fig. 13 over 1000 instances.

iterations. This feasible solution is the same as the one shown in Fig. 9, with network cost 10. By comparing Fig. 12 with Fig. 9, we can see that Algorithm 5 converges much more slowly than Algorithm 3. Fig. 13 illustrates the convergence of Algorithm 6. We can see that there exist for Problem 2, multiple feasible mixing solutions which have different network costs, and running Algorithm 5 for multiple times can result in different feasible solutions. Thus, the minimum network cost may decrease as the number of iterations increases. Algorithm 6 obtains the optimal network cost 10 to Problem 2 within 100 iterations. Fig. 14 illustrates the average convergence of Algorithm 6 over 1000 instances. We can see that on average, within 300 iterations, the minimum network cost under Algorithm 6 converges to 10, which is the optimal network cost to Problem 2 obtained by the centralized algorithm in Algorithm 1. By comparing Fig. 14 with Fig. 11, we can see that Algorithm 6 converges much more slowly than Algorithm 4.

VII. Conclusion and Future Work

In this paper, we introduce linear network mixing coefficients for code constructions of general integer connections. For such code constructions, we pose the problem of cost minimization for the subgraph involved in the coding solution, and relate this minimization to a path-based CSP and an edge-based CSP, respectively. We present a path-based probabilistic distributed algorithm and an edge-based probabilistic distributed algorithm with almost sure convergence in finite time by applying CFL. Our approach allows fairly general coding across flows, guarantees no greater cost than routing, and demonstrates a possible distributed implementation. Numerical results illustrate the performance improvement of our approach over existing methods.

This paper opens up several directions for future research. For instance, the proposed optimization-based linear network code construction for general integer connections can be extended to design route finding protocols of superior performance for general connections. In addition, a possible direction for future research is to design dynamic approaches not only to build new subgraphs, but also to update them as they evolve, so as to reflect changes in topologies for varying networks, as occur in such settings as peer-to-peer (P2P) networks. Another interesting extension of the proposed approach to content-centric cache-enabled networks is to incorporate cache placement (which creates new sources) into the cost minimization for the subgraph involved in the coding solution in this work. Finally, the proposed approach for wireline networks can also be generalized to wireless networks by considering hyper edges to model broadcast links.

References

[1] R. Koetter and M. Médard, "Beyond routing: an algebraic approach to network coding," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, October 2002, pp. 122–130 vol.1.
[2] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *Information Theory, IEEE Transactions on*, vol. 49, no. 2, pp. 371–381, February 2003.
[3] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, October 2006.
[4] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, September 2005, pp. 264–267.
[5] S. El Rouayheb, A. Sprintson, and C. Georghiades, "A new construction method for networks from matroids," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, June 2009, pp. 2872–2876.
[6] Q. Sun, S. T. Ho, and S.-Y. Li, "On network matroids and linear network codes," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, July 2008, pp. 1833–1837.
[7] R. Dougherty, C. Freiling, and K. Zeger, "Matroidal networks," in *Allerton Conference on Communication, Control, and Computing*, September 2007.
[8] A. Kim and M. Médard, "Scalar-linear solvability of matroidal networks associated with representable matroids," in *Turbo Codes and Iterative Information Processing (ISTC), 2010 6th International Symposium on*, Sep 2010, pp. 452–456.
[9] X. Yan, R. Yeung, and Z. Zhang, "The capacity region for multi-source multi-sink network coding," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, June 2007, pp. 116–120.
[10] T. Chan, A. Grant, and D. Pfluger, "Truncation technique for characterizing linear polymatroids," *Information Theory, IEEE Transactions on*, vol. 57, no. 10, pp. 6364–6378, 2011.
[11] V. T. Muralidharan and B. S. Rajan, "Linear index coding and representable discrete polymatroids," in *Information Theory (ISIT), 2014 IEEE International Symposium on*, June 2014, pp. 486–490.
[12] R. Dougherty, C. Freiling, and K. Zeger, "Networks, matroids, and non-shannon information inequalities," *Information Theory, IEEE Transactions on*, vol. 53, no. 6, pp. 1949–1969, June 2007.
[13] A. Kim and M. Médard, "Computing bounds on network capacity regions as a polytope reconstruction problem," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, July 2011, pp. 588–592.
[14] R. Dougherty, C. Freiling, and K. Zeger, "Achievable rate regions for network coding," in *Information Theory and Applications Workshop (ITA), 2012*, February 2012, pp. 160–167.
[15] S. El Rouayheb, A. Sprintson, and C. Georghiades, "On the index coding problem and its relation to network coding and matroid theory," *Information Theory, IEEE Transactions on*, vol. 56, no. 7, pp. 3187–3195, 2010.
[16] A. Salimi, M. Médard, and S. Cui, "On the representability of integer polymatroids: Applications in linear code construction," in *53rd Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, September 2015.
[17] W. Zeng, V. R. Cadambe, and M. Médard, "An edge reduction lemma for linear network coding and an application to two-unicast networks," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, 2012, pp. 509–516.
[18] C.-C. Wang and N. Shroff, "Pairwise intersession network coding on directed networks," *Information Theory, IEEE Transactions on*, vol. 56, no. 8, pp. 3879–3900, August 2010.
[19] S. Kamath, D. Tse, and V. Anantharam, "Generalized network sharing outer bound and the two-unicast problem," in *Network Coding (NetCod), 2011 International Symposium on*, July 2011, pp. 1–6.
[20] S. Kamath, V. Anantharam, D. N. C. Tse, and C. Wang, "The two-unicast problem," *CoRR*, vol. abs/1506.01105, 2015. [Online]. Available: http://arxiv.org/abs/1506.01105

[21] W. Zeng, V. R. Cadambe, and M. Médard, "A recursive coding algorithm for two-unicast-z networks," *http://www.mit.edu/ viveck/resources/ITW14twounicastz.pdf*.

[22] C. Meng, A. K. Das, A. Ramakrishnan, S. A. Jafar, A. Markopoulou, and S. Vishwanath, "Precoding-based network alignment for three unicast sessions," *arXiv preprint arXiv:1305.0868*, 2013.

[23] W. Zeng, V. Cadambe, and M. Médard, "On the tightness of the generalized network sharing bound for the two-unicast-z network," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, July 2013, pp. 3085–3089.

[24] H. Maleki, V. R. Cadambe, and S. A. Jafar, "Index coding-an interference alignment perspective," *arXiv preprint arXiv:1205.1483*, 2012.

[25] C. Li, S. Weber, and J. M. Walsh, "On multi-source networks: Enumeration, rate region computation, and hierarchy," *CoRR*, vol. abs/1507.05728, 2015. [Online]. Available: http://arxiv.org/abs/1507.05728

[26] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2608–2623, 2006.

[27] D. S. Lun, M. Médard, T. Ho, and R. Koetter, "Network coding with a cost criterion," in *in Proc. 2004 International Symposium on Information Theory and its Applications (ISITA 2004)*, 2004, pp. 1232–1237.

[28] Y. Wu, "On constructive multi-source network coding," in *Information Theory, 2006 IEEE International Symposium on*, July 2006, pp. 1349–1353.

[29] D. Traskov, N. Ratnakar, D. Lun, R. Koetter, and M. Médard, "Network coding for multiple unicasts: An approach based on linear optimization," in *Information Theory, 2006 IEEE International Symposium on*, July 2006, pp. 1758–1762.

[30] A. Khreishah, C.-C. Wang, and N. Shroff, "Optimization based rate control for communication networks with inter-session network coding," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[31] M. Kim, M. Médard, U.-M. O'Reilly, and D. Traskov, "An evolutionary approach to inter-session network coding," in *INFOCOM 2009, IEEE*, April 2009, pp. 450–458.

[32] Y. Cui, M. Médard, E. Yeh, D. Leith, and K. Duffy, "Optimization-based linear network coding for general connections of continuous flows," in *IEEE Int. Conf. on Commun. (ICC)*, London, UK, June 2015, pp. 4492–4498.

[33] Y. Cui, M. Medard, D. Pandya, E. Yeh, D. Leith, and K. Duffy, "A linear network code construction for general integer connections based on the constraint satisfaction problem," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–7.

[34] K. Duffy, C. Bordenave, and D. Leith, "Decentralized constraint satisfaction," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 4, pp. 1298–1308, August 2013.

[35] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765–1775, October 2011.

[36] C. Fragouli and E. Soljanin, "Network coding fundamentals," *Found. Trends Netw.*, vol. 2, no. 1, pp. 1–133, January 2007.

Appendix A: Proof of Theorem 1

Let $\mathbf{z}$, $\mathbf{x}$, $\boldsymbol{\beta}$ and $\mathbf{f}$ denote a feasible solution to Problem 1. Note that $\mathbf{x}$ is uniquely determined by $\boldsymbol{\beta}$ according to (10) and (11), which correspond to Conditions 1) and 2) in Definition 2. In addition, by (12), which corresponds to Condition 3) in Definition 2, we know that $\mathbf{x}$ ensures that for each terminal, the extraneous flows are not mixed with the desired flows on the paths to the terminal. We shall show that, based on $\boldsymbol{\beta}$, we can find local coding coefficients $\boldsymbol{\alpha}$, which uniquely determine feasible global coding coefficients $\mathbf{c}$ according to

$$\mathbf{c}_{s_p j} = \mathbf{e}_p, \quad (s_p, j) \in \mathcal{E}, \ p \in \mathcal{P} \tag{28}$$

$$\mathbf{c}_{ij} = \sum_{k \in \mathcal{I}_i} \alpha_{kij} \mathbf{c}_{ki}, \quad (i,j) \in \mathcal{E}, i \notin \mathcal{S} \tag{29}$$

where correspond to Conditions 1) and 2) in Definition 1).

First, we choose $\alpha_{kij} = 0$ if $\beta_{kij} = 0$. Note that, as a feasible solution, $\mathbf{x}$ is uniquely determined by $\boldsymbol{\beta}$ according to (10) and (11). In addition, we choose $\mathbf{c}$ based on $\boldsymbol{\alpha}$ according to (28) and (29). Thus, by (28), (29), (10) and (11), we can show that $c_{ij,p} = 0$ if $x_{ij,p} = 0$ by induction. Thus, by (12), we have

$$c_{it,p} = 0, \quad i \in \mathcal{I}_t, \ p \notin \mathcal{P}_t, \ t \in \mathcal{T}. \tag{30}$$

In other words, each terminal $t \in \mathcal{T}$ only needs to consider $(c_{it,p})_{i \in \mathcal{I}_t, p \in \mathcal{P}_t}$ for decoding. By (8), we can form a flow path from source s_p to terminal t, which consists of the edges in $\mathcal{L}_p^t \triangleq \{(i,j) \in \mathcal{E} : f_{ij,p}^t = 1\}$, where $p \in \mathcal{P}_t$. By (3) and (7), we know that for all $t \in \mathcal{T}$, there exists P_t edge-disjoint unit flow paths, each one from one source s_p to terminal t, where $p \in \mathcal{P}_t$. Note that $\mathbf{x}$ satisfies all the conditions in Definition 2. Thus, by (9), we know that all the flow paths satisfy that for each terminal, the extraneous flows (information) are not mixed with the desired flows (information) on the flow paths to the terminal. Let A_t denote the $P_t \times P_t$ matrix, each row (out of P_t rows) of which consists of the P_t elements in $(c_{it,p})_{p \in \mathcal{P}_t}$ for the last edge (i,t) on one flow path (out of P_t flow paths) to terminal t, where $i \in \mathcal{I}_t$. Note that A_t (in terms of $(c_{it,p})_{i \in \mathcal{I}_t, p \in \mathcal{P}_t}$ for all P_t flow paths) can also be expressed in terms of local coding coefficients $\boldsymbol{\alpha}$ by (28) and (29).[12] By (28) and (29), we know that 1) and 2) of Definition 1 are satisfied. Therefore, it remains to show that 3) of Definition 1 is satisfied. This can be achieved by choosing $\{\alpha_{kij} : (k,i),(i,j) \in \mathcal{E}, \beta_{kij} \neq 0\}$ so that A_t for all $t \in \mathcal{T}$ are full rank, i.e., $\prod_{t \in \mathcal{T}} \det(A_t) \neq 0$ [36, Pages19-20].

Next, we show that if $F > T$, we can choose $\{\alpha_{kij} : (k,i),(i,j) \in \mathcal{E}, \beta_{kij} \neq 0\}$ such that $\prod_{t \in \mathcal{T}} \det(A_t) \neq 0$. We first show that for all $t \in \mathcal{T}$, $\det(A_t)$ is not identically equal to zero. For all $p \in \mathcal{P}_t$ and $t \in \mathcal{T}$, choose $\alpha_{kij} = 1$ for all edges $(k,i),(i,j) \in \mathcal{E}$ on the flow path from source s_p to terminal t, i.e., $(k,i),(i,j) \in \mathcal{L}_p^t$, and $\alpha_{kij} = 0$ for all edges $(k,i),(i,j) \in \mathcal{E}$ not on the same flow path, i.e., (k,i) or $(i,j) \notin \mathcal{L}_p^t$. This local coding coefficient assignment makes A_t the $P_t \times P_t$ identity matrix. Thus, $\det(A_t)$ is not identically equal to zero [36, Page 20]. Then, we show that $\prod_{t \in \mathcal{T}} \det(A_t)$ is not equal to zero, using the algebraic framework in [36, Pages 31-32]. Similarly to the proof of Theorem 3.2 in [36, Pages 31-32], we can show that $\prod_{t \in \mathcal{T}} \det(A_t)$ is a polynomial in unknown variables $\{\alpha_{kij} : (k,i),(i,j) \in \mathcal{E}, \beta_{kij} \neq 0\}$ and that the degree of each unknown variable is at most T. Therefore, by Lemma 2.3 [36, Page 21], we can show that, for $F > T$, there exists a choice of $\{\alpha_{kij} : (k,i),(i,j) \in \mathcal{E}, \beta_{kij} \neq 0\}$ such that $\prod_{t \in \mathcal{T}} \det(A_t) \neq 0$. Recall that $\alpha_{kij} = 0$ if $\beta_{kij} = 0$. Therefore, based on $\boldsymbol{\beta}$, we can obtain $\boldsymbol{\alpha}$ that leads to feasible $\mathbf{c}$.

[12]Given all the local coding coefficients $\boldsymbol{\alpha}$, we can compute global coding coefficients $\mathbf{c}$, and vice versa.