The Approximate Capacity of Half-Duplex Line Networks

Yahya H. Ezzeldin, Martina Cardone, Christina Fragouli and Daniela Tuninetti

***Abstract*—This paper investigates the problem of characterizing the capacity of Half-Duplex (HD) *line networks*, where a source node communicates to a destination node through a multi-hop path of N relays. If the relays operate in Full-Duplex (FD), it is well known that the capacity of the line network equals the minimum among the point-to-point link capacities in the path. In contrast, this paper considers a different case where the relays operate in HD. In the first part of the paper, it is shown that the approximate capacity (optimal up to a constant additive gap that only depends on the number of nodes in the network) of an HD N-relay line network equals half the minimum of the harmonic means of the point-to-point link capacities of each two consecutive links in the path. It is then proved that the $N+1$ listen/transmit states (out of the 2^N possible ones) sufficient to characterize the approximate capacity can be found in linear time. In the second part of the paper, it is shown that the problem of finding the path that has the largest HD approximate capacity in a network that can be represented as a graph is NP-hard. However, if the number of cycles in the network is polynomial in the number of nodes, then a polynomial-time algorithm can indeed be designed.**

I. Introduction

In recent years, promising advances have been made in designing Full-Duplex (FD) transceivers [1], [2]. However, the proposed FD designs still require complex self-interference cancellation techniques. Due to this, in the near future it is envisioned that nodes will continue to operate in Half-Duplex (HD) mode in order to enable low-cost communications – as recently announced, for example, in 3GPP Rel-13 [3]. A widespread approach to route information from a source node to a destination node in an HD network is to find the path with the largest FD capacity and then operate the path in HD mode [4], [5], [6]. This approach is used because of the simple nature of the FD capacity expression, which is given by the minimum of the point-to-point link capacities in the path, and thus can be distributively computed in linear time. However, selecting a route based on its FD capacity may be suboptimal if then the nodes in the selected path are operated in HD mode.

In this paper we investigate the problem of routing in HD networks. We address the three following fundamental questions: (i) Can a closed-form expression of the capacity (or of an approximation of it) of an N-relay HD line network be derived with the same promising features of the FD counterpart? (ii) Can the linear number of active listen/transmit configuration states sufficient to characterize the HD capacity (or an approximation of it) be found efficiently in polynomial-time? (iii) Does there exist a low-complexity algorithm that finds the route in a network with the largest HD capacity (or an approximation of it)?

A. Related Work

A route/path connecting a source node to a destination node through N relays is an N-relay line network. Since the line network is a physically degraded relay channel, its capacity is known to be given by the cut-set upper bound [7] and achieved by decode-and-forward. While for the FD case the capacity can be expressed in an elegant and simple form as the minimum among the point-to-point link capacities in the line network, a similar result for the HD case is not yet available, to the best of our knowledge. The main reason for this is due to the fact that in HD the channel input at each relay is also characterized by the state (either listen or transmit) of that particular relay [8]. This allows to transmit further information from the source to the destination by switching among the 2^N possible listen/transmit states that can occur inside the network. Given this, it is not clear what is the optimal input distribution that maximizes the cut-set upper bound. Recent results in [9], [10], [11] generalized an observation in [8], by showing that the HD capacity of a Gaussian relay network can be approximated to within an additive constant gap (i.e., which is independent on the channel parameters and only depends on the number of relays N) by the cut-set upper bound evaluated with a deterministic schedule independent of the transmitted and received signals and with independent inputs. Throughout the paper, we refer to this as the *approximate capacity*.

For the class of relay networks defined in [12, Theorem 1] (which includes the practically relevant Gaussian noise case), the evaluation of the approximate capacity of an HD relay network can be cast as an optimization problem over 2^N cuts of the network (as in FD) each of which is a function of 2^N listen/transmit configuration states. As N increases,

Y. H. Ezzeldin and C. Fragouli are with the Electrical and Computer Engineering Department at the University of California, Los Angeles, CA 90095 USA (e-mail: {yahya.ezzeldin, christina.fragouli}@ucla.edu). Their research was partially funded by NSF under award numbers 1514531 and 1824568. M. Cardone is currently with the Electrical and Computer Engineering Department of the University of Minnesota, MN 55404 USA (e-mail: cardo089@umn.edu). At UCLA, M. Cardone was partially supported by NSF under award number 1314937. D. Tuninetti is with the Electrical and Computer Engineering Department of the University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: danielat@uic.edu). The work of D. Tuninetti was partially funded by NSF under award number 1527059.

The results in this paper were presented in part at the 2017 IEEE International Symposium on Information Theory and at the 55th Annual Allerton Conference on Communication, Control, and Computing.

this evaluation, as well as determining an optimal schedule, become computationally expensive. In [12] the authors proved the conjecture posed in [13] in the context of Gaussian diamond networks[1], which states that at most $N+1$ states are sufficient for approximate capacity characterization for a class of HD N-relay networks, which includes the Gaussian noise case. A schedule with at most $N+1$ active states is referred to as *simple*. This result is promising since it implies that the network can be operated close to its capacity with a limited number of state switches. However, to the best of our knowledge, it is not clear yet if such simple schedules can be found with low-complexity algorithms. The authors in [14] designed an iterative algorithm to determine a schedule optimal for the approximate capacity when the relays employ decode-and-forward. In [15], the authors proposed a node-grouping technique that provides polynomial-time algorithms to compute the approximate capacity of certain classes of Gaussian HD relay networks that include the line network as special case. While the results in [14], [15] show that the approximate capacity can be found in polynomial-time for special network topologies by solving a linear program (LP), it is not clear how to efficiently construct a schedule that achieves it. In contrast, in this work we derive the approximate capacity of a general memoryless HD line network in closed form and we design an efficient algorithm, which outputs in $O(N)$ time a simple schedule that achieves it.

In [16], the exact capacity region is derived for a line network where the source node, and possibly multiple relays along the way, have messages intended to the destination. The capacity in [16] is given as a maximization over the augmented alphabet used for the channel inputs (the alphabet used in the transmitted signal plus an additional random variable denoting that a node is listening). Additionally, the optimization in [16] does not explicitly give the distribution for the channel inputs and does not provide an explicit schedule for the network. In particular, given this formulation, it is possible to use the schedule to send additional information; this follows since the schedule can now be considered part of the distribution of the channel input [8]. In contrast, in our work, we consider a noisy line network model, where the point-to-point link capacities can be different from one link to the next (implying that different alphabets are used for transmission over each link). Furthermore, we upper bound the amount of information that can be transfered through the schedule by a quantity that only depends on N, and we look at the approximate capacity, where a fixed schedule is selected based on the network parameters but is not part of the codebook. This allows us to develop a closed-form expression for the approximate capacity.

Given the derived closed-form expression for the approximate capacity of an HD line network, we then analyze the problem of HD routing, which consists of efficiently discovering the path with the largest HD approximate capacity in a relay network. As a result of its widespread use in currently deployed wireless networks, routing has been extensively studied in the literature. For instance, a line of work [17], [18], [19] focused on finding a route between the source and the destination by assuming that the point-to-point link capacities can only take values of 0 or 1. Under this assumption, the route selection based on the FD point-to-point link capacities is optimal. Finding the route with the largest FD capacity is equivalent to the problem of finding the widest-path between a pair of vertices in a graph [20]. This can be efficiently solved by adapting any algorithm that finds the shortest path between a pair of vertices in a graph (e.g., Dijkstra's algorithm [21]). For routing with multi-rate link capacities, several heuristic metrics were proposed to enhance the selection of routes in ad-hoc wireless networks [4], [5]. In contrast with this set of works, we are interested in selecting the route with the largest HD approximate capacity, by also trying to address the fundamental complexity of finding such a route.

[1]An N-relay diamond network is a relay network topology where the source can communicate with the destination only through N non-interfering relays.

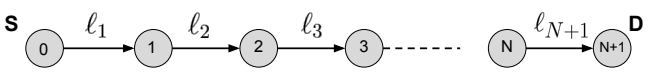

Fig. 1. The line network $\mathcal{R}$ with N relays, source S and destination D.

B. Contributions

In this paper, we study the problem of characterizing the capacity of N-relay HD line networks. We also study the implications of this result on the problem of routing in HD networks, where the goal is to find the route from a source to a destination with the largest HD approximate capacity. Our main contributions can be summarized as follows:

1) We derive the HD approximate capacity of the N-relay line network $\mathcal{R}$ (shown in Fig. 1) in closed form and show it is given by half of the minimum of the harmonic means of the point-to-point capacities of each two consecutive links in the path, that is
$$\mathsf{C}_{\mathcal{R}} = \min_{i\in[1:N]} \left\{ \frac{\ell_i \ell_{i+1}}{\ell_i + \ell_{i+1}} \right\}, \tag{1}$$
where ℓ_i is the point-to-point capacity of the link between node $i-1$ and node i. This approximate capacity expression has the same appealing features of the FD counterpart, i.e., it can be evaluated in linear time and distributively computed among the nodes in the path. To the best of our knowledge, this is the first approximate capacity characterization in closed form for a class of HD relay networks with general number of relays.
2) We prove that, with only the knowledge of the network topology (i.e., that the N relays are arranged in a line), the cardinality of the smallest search space of states over which a schedule that achieves the approximate capacity can be found is exponential in N. In other words, to reduce the cardinality of this search space to be polynomial in the number of relays, it is crucial to leverage the strength of the channel parameters.
3) We design an algorithm that allows to compute a simple schedule (i.e., with at most $N+1$ active states) that achieves the approximate capacity of the N-relay HD line network with complexity $O(N)$. This result sheds

TABLE I
QUANTITIES OF INTEREST USED THROUGHOUT THE PAPER.

Quantity	Definition
$\mathcal{G}$	Digraph representing a relay network
$\mathcal{R}$	Line network
$\mathcal{L}_{\mathcal{G}}$	Line digraph of the digraph $\mathcal{G}$
$\mathcal{P}$	Path between two nodes
$\ell_{i,j}$	Point-to-point link capacity from node v_i to node v_j in $\mathcal{G}$
ℓ_i	Point-to-point link capacity from node v_{i-1} to node v_i in $\mathcal{R}$
$\mathsf{C}_{\mathcal{R}}$ (resp. $\mathsf{C}_{\mathcal{P}}$)	Approximate HD capacity of $\mathcal{R}$ (resp. $\mathcal{P}$)
$\mathsf{C}_{\mathcal{R}}^{\lambda}$	Achievable HD rate for $\mathcal{R}$ with deterministic schedule λ
$\mathsf{C}_{\mathcal{R}}^{\mathrm{FD}}$ (resp. $\mathsf{C}_{\mathcal{P}}^{\mathrm{FD}}$)	FD capacity of $\mathcal{R}$ (resp. $\mathcal{P}$)

light on how to operate a class of HD relay networks close to the capacity with the minimum number of state switches. Moreover, to the best of our knowledge, this is the first result that provides an efficient way to find a simple schedule optimal for approximate capacity.

4) We prove that the problem of finding the route with the largest HD approximate capacity in a relay network is NP-hard in general. Our proof is based on a reduction from the 3SAT problem [22], which is a special case of the well-known SAT problem, where the goal is to determine the satisfiability of a Boolean formula. The NP hardness of HD routing represents a surprising difference from FD, for which polynomial-time algorithms exist to discover the path with the largest FD capacity, such as Dijkstra's algorithm [21]. Intuitively, the NP-hardness in HD routing stems from the necessity to avoid cycles in the network while discovering an HD path, which is not necessary when finding a FD path.
5) We show that, if the number of cycles in the network is polynomial in the total number of nodes, then a polynomial-time algorithm that discovers the path with the largest HD approximate capacity can be designed. Thus, this represents a sufficient condition for which HD routing can be efficiently solved. A relevant class of relay networks for which this holds is the one of layered networks where the relays are arranged over L layers of relays and a relay in a layer can only communicate to the relays in the next layer.

C. Paper Organization

Section II illustrates the problem setting of our problem, describes known capacity results for HD line networks and simplifies the approximate capacity expression for HD line networks. Section III presents our main results and discusses their implications. Section IV proves the NP-hardness of finding the route with the largest HD approximate capacity in a relay network. Section V describes special network classes for which a polynomial-time algorithm for finding the route with the largest HD approximate capacity exists. Section VI concludes the paper. Some of the proofs are delegated to the Appendix.

II. HALF-DUPLEX NETWORK MODEL

Table I summarizes and defines quantities that are used throughout the paper.

We consider an HD relay network represented by the directed graph $\mathcal{G}$ where $\mathcal{V}(\mathcal{G})$ and $\mathcal{E}(\mathcal{G})$ are the set of vertices (communication nodes) and the set of edges (point-to-point links) in $\mathcal{G}$, respectively. The point-to-point links between nodes in the network are assumed to be non-interfering discrete memoryless channels. An edge connecting vertex v_i to vertex v_j where $v_i, v_j \in \mathcal{V}(\mathcal{G})$ is denoted by $e_{i,j}$. For each edge $e_{i,j} \in \mathcal{E}(\mathcal{G})$, we represent its point-to-point link capacity with $\ell_{i,j} > 0$. Over the graph $\mathcal{G}$ with $N+2$ vertices, information flows from a source node $S \in \mathcal{V}(\mathcal{G})$ (denoted by v_0) to a destination node $D \in \mathcal{V}(\mathcal{G})$ (denoted by v_{N+1}) with the help of the remaining N relay nodes. Each node in $\mathcal{G}$ operates in HD, i.e., it cannot transmit and receive simultaneously.

A relay network is called a *line network* if its vertices are arranged in a path (or a route) forming a cascade of non-interfering discrete memoryless channels[2]. The input/output relation for the line network (denoted by $\mathcal{R}$) with N relays can therefore be defined through the conditional distribution

$$p\Big(Y_1,\ldots,Y_{N+1}\Big|\{(X_0,S_0)\}_{i=0^N},S_{N+1}\Big) = \prod_{i=0}^{N} p\Big(Y_{i+1}\Big|(X_i,S_i),S_{i+1}\Big), \tag{2}$$

where: (i) X_i (respectively, Y_i) denotes the channel input (respectively, output) at node v_i; (ii) S_i is the binary random variable which represents the state of node v_i, i.e., if $S_i = 0$ then node v_i is receiving, while if $S_i = 1$ then node v_i is transmitting; notice that $S_0 = 1$ (i.e., the source always transmits) and $S_{N+1} = 0$ (i.e., the destination always receives). The line network in (2) is a cascade of N discrete memoryless channels and as a result is physically degraded [7]. Thus, the capacity of the line network $\mathcal{R}$ is given by the cut-set bound that can be achieved by decoding transmissions from node i at node $i+1$ before encoding them for transmission further in the network as would typically happen with routing. This is exactly how the standard decode-and-forward relaying scheme operates over the line network. In particular, the capacity is given by the cut-set bound as in (3), at the top of the next page, where $\mathcal{A}$ represents the set of relays on the destination side of the cut, and $\mathcal{A}^c = [1{:}N]\backslash\mathcal{A}$. However, it is not clear

[2]A line network consists of a cascade of noisy channels, which make it a physically degraded channel. Therefore, a node has a "cleaner" view of information with respect to a node that is next in the line and thus, replacing directed edges with undirected ones does not increase the capacity.

$$\mathsf{C}_{\mathcal{R}}^{(\mathrm{cs})} = \max_{p(X_0,\{X_i,S_i\}_{i=1}^N)} \min_{\mathcal{A}\subseteq[1:N]} I(Y_{N+1},\{Y_i\}_{i\in\mathcal{A}}; X_0,\{X_i,S_i\}_{i\in\mathcal{A}^c}|\{X_i,S_i\}_{i\in\mathcal{A}}), \tag{3}$$

what is the optimal distribution of $\{(X_i,S_i)\}_{i=0}^N$ needed to characterize the capacity of the HD line network $\mathcal{R}$ in (3).

The capacity of the HD line network $\mathcal{R}$ described in (2) can however be approximated to within a constant gap $\mathrm{GAP} = N$ by using deterministic schedules. In particular, to obtain this constant gap approximation we upper bound the cut-set bound in (3) as seen in (4), at the top of the next page, where: (a) follows from the chain rule of the mutual information and by the fact that, for a discrete random variable, the entropy is a non-negative quantity (the mutual information can hence be upper bounded by the entropy); and (b) follows by upper bounding the entropy of a discrete random variable by the logarithm of its support.

This constant gap capacity approximation, which is derived using similar arguments as in [9], [10], [11] for Gaussian HD relay networks, follows since a binary random variable (i.e., a relay state) can only improve the capacity by at most 1 bit. Hence, since we have N relays, the gap is at most equal to N. The first term $\mathsf{C}_{\mathcal{R}}$ in (4) which uses fixed schedules (i.e., the exact values of $\{S_i\}$ are in the conditioning of the mutual information term) is what we refer to as the *approximate capacity*. By using decode-and-forward with an optimal product input distribution and deterministic schedules, the approximate capacity can be achieved and is expressed by

$$\mathsf{C}_{\mathcal{R}} = \max_{\lambda\in\Lambda} \min_{\mathcal{A}\subseteq[1:N]} \sum_{s\in[0:1]^N} \lambda_s \sum_{\substack{i\in\{N+1\}\cup\{\mathcal{T}_s^c\cap\mathcal{A}\}\\ i-1\in\{0\}\cup\{\mathcal{T}_s\cap\mathcal{A}^c\}}} \ell_i, \tag{5}$$

where: (i) the schedule $\lambda\in\mathbb{R}^{2^N}$ determines the fraction of time the network operates in each of the states $s\in[0:1]^N$, i.e., $\lambda_s = \Pr\left(\{S_i\}_{i=1}^N = s\right)$; (ii) $\Lambda = \left\{\lambda : \lambda\in\mathbb{R}^{2^N},\ \lambda\geq 0,\ \sum_{s\in[0:1]^N}\lambda_s = 1\right\}$ is the set of all possible schedules; (iii) $\mathcal{T}_s$ (respectively, $\mathcal{T}_s^c = [1:N]\backslash\mathcal{T}_s$) represents the set of indices of relays transmitting (respectively, receiving) in the state $s\in[0:1]^N$; (iv) for ease of notation, we set $\ell_i = \ell_{i-1,i}$ to denote the point-to-point capacity of the link from v_{i-1} to v_i. We can equivalently write the expression in (5) as

$$\mathsf{C}_{\mathcal{R}} = \max_{\lambda\in\Lambda} \min_{\mathcal{A}\subseteq[1:N]} \sum_{s\in[0:1]^N} \lambda_s \sum_{\substack{i\in\{N+1\}\cup\mathcal{A}\\ i-1\in\{0\}\cup\mathcal{A}^c}} \hat{\ell}_i^{(s)}, \tag{6}$$

where

$$\hat{\ell}_i^{(s)} := \begin{cases} \ell_i, & \text{if } i\in\mathcal{T}_s^c\cup\{N+1\} \text{ and } i-1\in\mathcal{T}_s\cup\{0\}\\ 0, & \text{otherwise.}\end{cases} \tag{7}$$

Similarly, we denote with $\mathsf{C}_{\mathcal{R}}^{\lambda}$, the HD rate achieved by the line network $\mathcal{R}$ when operated with the deterministic schedule λ, i.e.,

$$\mathsf{C}_{\mathcal{R}}^{\lambda} = \min_{\mathcal{A}\subseteq[1:N]} \sum_{s\in[0:1]^N} \lambda_s \sum_{\substack{i\in\{N+1\}\cup\mathcal{A}\\ i-1\in\{0\}\cup\mathcal{A}^c}} \hat{\ell}_i^{(s)}. \tag{8}$$

Note that, for all possible schedules λ, $\mathsf{C}_{\mathcal{R}}^{\lambda}\leq\mathsf{C}_{\mathcal{R}}$.

Definition 1 (Simple Schedule)**.** We say that a schedule $\lambda\in\mathbb{R}^{2^N}$ is *simple* if the number of active states, i.e., states s such that $\lambda_s > 0$ is at most $N+1$. In other words, λ is simple if $\|\lambda\|_0\leq N+1$ (with $\|\lambda\|_0$ being the L0 norm of the vector λ). In [12, Theorem 1], it was shown that for any Gaussian HD relay network with arbitrary topology, there always exists a simple schedule that is optimal for the approximate capacity.

A. Fundamental Cuts in HD Line Networks

In this subsection, we prove that for the HD line network in (2), we can compute $\mathsf{C}_{\mathcal{R}}$ in (6) by considering only $N+1$ cuts (out of the 2^N possible ones), which are the same that one would consider if the network was operating in FD.

For the line network $\mathcal{R}$, when all the N relays operate in FD, the FD capacity is given by

$$\mathsf{C}_{\mathcal{R}}^{\mathrm{FD}} = \min_{\mathcal{A}\subseteq[1:N]} \sum_{\substack{i\in\{N+1\}\cup\mathcal{A},\\ i-1\in\{0\}\cup\mathcal{A}^c}} \ell_i = \min_{i\in[1:N+1]} \{\ell_i\}, \tag{9}$$

that is, without *explicit* knowledge of the values of ℓ_i or their ordering, the number of cuts over which we need to optimize (see $\mathsf{C}_{\mathcal{R}}^{\mathrm{FD}}$ in (9)) is $N+1$. We refer to these cuts as *fundamental*. When states or cuts are referred to as fundamental of a certain type (e.g., maximum, minimum), we mean that they form the smallest set of that type that only depends on the network topology (i.e., relays are arranged in a line) and is independent of the actual values of the point-to-point link capacities. Let $\mathscr{F}$ denote the set of these fundamental cuts (which are of the form $\mathcal{A} = [i:N], i\in[1:N]$ or $\mathcal{A}=\emptyset$). For any cut $\mathcal{A}$ of the network

$$\sum_{\substack{i\in\{N+1\}\cup F(\mathcal{A}),\\ i-1\in\{0\}\cup F(\mathcal{A})^c}} \ell_i \leq \sum_{\substack{i\in\{N+1\}\cup\mathcal{A},\\ i-1\in\{0\}\cup\mathcal{A}^c}} \ell_i \quad \text{for some } F(\mathcal{A})\in\mathscr{F}. \tag{10}$$

Furthermore, the function $F(\cdot)$ in (10) does not depend on the values of ℓ_i.

We next prove that the fundamental cuts in HD equal those in (9) for FD. Consider a deterministic schedule λ. Then, by using (10) for the inner summation in (8), for each $s\in[0:1]^N$ we have

$$\sum_{\substack{i\in\{N+1\}\cup F(\mathcal{A}),\\ i-1\in\{0\}\cup F(\mathcal{A})^c}} \hat{\ell}_i^{(s)} \leq \sum_{\substack{i\in\{N+1\}\cup\mathcal{A},\\ i-1\in\{0\}\cup\mathcal{A}^c}} \hat{\ell}_i^{(s)}. \tag{11}$$

Thus, we can simplify (8) as

$$\begin{aligned}\mathsf{C}_{\mathcal{R}}^{\lambda} &= \min_{\mathcal{A}\subseteq[1:N]} \sum_{s\in[0:1]^N} \lambda_s \sum_{\substack{i\in\{N+1\}\cup\mathcal{A}\\ i-1\in\{0\}\cup\mathcal{A}^c}} \hat{\ell}_i^{(s)}\\ &= \min_{\mathcal{A}\in\mathscr{F}} \sum_{s\in[0:1]^N} \lambda_s \sum_{\substack{i\in\{N+1\}\cup\mathcal{A}\\ i-1\in\{0\}\cup\mathcal{A}^c}} \hat{\ell}_i^{(s)}\end{aligned}$$

$$\begin{aligned}\mathsf{C}_{\mathcal{R}}^{(\mathrm{cs})} &= \max_{p(X_0,\{X_i,S_i\}_{i=1}^N)} \min_{\mathcal{A}\subseteq[1:N]} I(Y_{N+1},\{Y_i\}_{i\in\mathcal{A}};X_0,\{X_i,S_i\}_{i\in\mathcal{A}^c}|\{X_i,S_i\}_{i\in\mathcal{A}}) \\ &\overset{(a)}{\leq} \max_{p(X_0,\{X_i,S_i\}_{i=1}^N)} \min_{\mathcal{A}\subseteq[1:N]} I(Y_{N+1},\{Y_i\}_{i\in\mathcal{A}};X_0,\{X_i\}_{i\in\mathcal{A}^c}|\{X_i\}_{i\in\mathcal{A}},\{S_i\}_{i=1}^N) + H(\{S_i\}_{i=1}^N) \\ &\overset{(b)}{\leq} \underbrace{\max_{p(\{X_i\}_{i=0}^N|\{S_i\})p(\{S_i\}_{i=1}^N))} \min_{\mathcal{A}\subseteq[1:N]} I(Y_{N+1},\{Y_i\}_{i\in\mathcal{A}};X_0,\{X_i\}_{i\in\mathcal{A}^c}|\{X_i\}_{i\in\mathcal{A}},\{S_i\}_{i=1}^N)}_{\mathsf{C}_{\mathcal{R}}} + \underbrace{N}_{\mathrm{GAP}}, \end{aligned} \tag{4}$$

$$= \min_{i\in[1:N+1]} \left(\sum_{s\in\mathcal{S}_i} \lambda_s\right) \ell_i, \tag{12}$$

where

$$\mathcal{S}_i = \left\{s\in[0{:}1]^N | i\in\{N{+}1\}\cup\mathcal{T}_s^c,\ i{-}1\in\{0\}\cup\mathcal{T}_s\right\}. \tag{13}$$

The set $\mathcal{S}_i \subseteq [0:1]^N$ represents the collection of states that activate the i-th link. For illustration, for a network with $N = 3$ we have

$$\begin{aligned}\mathcal{S}_1 &= \{000,\ 001,\ 010,\ 011\},\\ \mathcal{S}_2 &= \{100,\ 101\},\\ \mathcal{S}_3 &= \{010,\ 110\},\\ \mathcal{S}_4 &= \{001,\ 011,\ 101,\ 111\}.\end{aligned}$$

Using the same arguments as in (12), we can similarly simplify the expression of $\mathsf{C}_{\mathcal{R}}$ in (6). Thus, the result presented in this section explicitly provides the $N+1$ cuts (out of the 2^N possible ones) over which it is sufficient to minimize in order to obtain $\mathsf{C}_{\mathcal{R}}$ in (6).

III. Main Results and Discussion

In this section, we present our main results and discuss their implications. Our first main result, stated in Theorem 1 is two-fold: (i) it provides a closed-form expression for the approximate capacity of the HD line network that can be evaluated in linear time, and (ii) it shows the existence of a polynomial-time algorithm that outputs a simple schedule optimal for approximate capacity.

Theorem 1. *For the N-relay HD line network $\mathcal{R}$ described in* (2)*, a simple schedule (i.e., with at most $N+1$ active states) optimal for approximate capacity can be obtained in $O(N)$ time and the approximate capacity $\mathsf{C}_{\mathcal{R}}$ in* (6) *is given by* (1)*.*

Proof: It is not difficult to argue that the right-hand side of (1) is an upper bound on $\mathsf{C}_{\mathcal{R}}$. This can be seen by assuming that, for a given $i\in[1:N]$, node v_{i-1} perfectly cooperates with node v_0 and node v_{i+1} perfectly cooperates with node v_{N+1} (see also Fig. 2 for an illustrative example with $i=2$ and $N=3$). Clearly, the HD approximate capacity of this new line network (equivalent to a single relay line network) is an upper bound on $\mathsf{C}_{\mathcal{R}}$ and is given by $\max_{0\leq\beta\leq1} \min\{(1-t_i)\ell_i\ , t_i\ell_{i+1}\} = \frac{\ell_i\ \ell_{i+1}}{\ell_i+\ell_{i+1}}$, which is achieved by setting $t_i = \frac{\ell_i}{\ell_i+\ell_{i+1}}$. Since this is true for all $i\in[1:N]$, then $\mathsf{C}_{\mathcal{R}}$ is less than or equal to the right-hand side of (1).

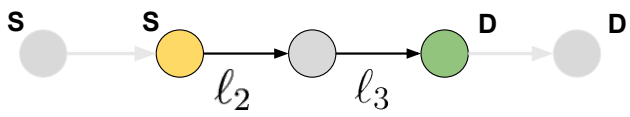

Fig. 2. Upper bound on $\mathsf{C}_{\mathcal{R}}$ for $i=2$.

To prove the achievability of (1), we assign a duration of time – denoted as TX_i – for each relay $i\in[1:N]$ to be transmitting (and hence listening in the remaining time). The transmit period TX_i assigned to relay i is parameterized by t_i and is given by the following period assignment

$$\mathrm{TX}_i = \begin{cases}[0,t_i], & i \text{ is even},\\ [1-t_i,1], & i \text{ is odd},\end{cases} \tag{14}$$

where $t_i = \dfrac{\ell_i}{\ell_i+\ell_{i+1}}, \forall i\in[1:N]$. We denote the time spent by relay i listening as $\mathrm{RX}_i = [0,1]\backslash\mathrm{TX}_i$.

It is not difficult to see that, with this time allocation, the network changes its state in at most N points in time given by the values $t_i, \forall i\in[1:N]$. Thus the proposed schedule has at most $N+1$ states. Furthermore, the schedule can be created in $O(N)$. What remains to show is that the proposed schedule achieves the rate given in (1). Towards this end, we need to compute the duration of time $\gamma_i, i\in[1:N]$, for which the link of capacity ℓ_i is active. This can be computed as follows

$$\begin{aligned}\gamma_i &= |\mathrm{TX}_{i-1}\cap\mathrm{RX}_i| \\ &\overset{(a)}{=} \begin{cases}\frac{\ell_2}{\ell_1+\ell_2}, & i=1,\\ \min\left(\frac{\ell_{i+1}}{\ell_i+\ell_{i+1}}, \frac{\ell_{i-1}}{\ell_i+\ell_{i-1}}\right), & i\in[2:N],\\ \frac{\ell_N}{\ell_{N+1}+\ell_N}, & i=N+1,\end{cases}\end{aligned} \tag{15}$$

where (a) follows by: (i) computing the size of the intersection $\mathrm{TX}_{i-1}\cap\mathrm{RX}_i$, (ii) using the ranges given in (14), and (iii) the fact that the source is always transmitting and the destination is always listening. The rate achieved by the given schedule is hence equal to $\mathsf{C}_{\mathcal{R}}^{\lambda} = \min_{i=[1:N+1]}\{\gamma_i\ell_i\}$, which gives the result in (1). This concludes the proof of Theorem 1. ∎

Remark 1. Although the described achievable scheme used for the proof of Theorem 1 yields a rate equal to the approximate capacity in (1), it can be over-using non-bottleneck links in the network. In particular, as described in the proof of Theorem 1, the i-th link has an effective scheduled rate of $\min\{\frac{\ell_i\ell_{i-1}}{\ell_i+\ell_{i-1}}, \frac{\ell_i\ell_{i+1}}{\ell_i+\ell_{i+1}}\}$, i.e., the effective used rate can be strictly greater than the network approximate capacity.

In [23], we proposed an alternative approach for scheduling the relays which is based on edge-coloring and we proved that it also achieves the approximate capacity in (1) by using the link of capacity ℓ_j with an effective scheduled rate of $\min_{j\in[1:N]}\{\frac{\ell_j\ell_{j+1}}{\ell_j+\ell_{j+1}}\}$. In other words, the edge-coloring algorithm proposed in [23] ensures that each link is active only for a duration of time that suffices to achieve the approximate capacity. This property of edge-coloring has been recently leveraged to develop queue-aware scheduling schemes [24] (assuming that the queues at the nodes are not infinite) that achieve rates that approach the approximate capacity. □

In what follows, we highlight some remarks to motivate the need to search for a simple schedule for the line network and to explain why our search and schedule presented in the proof of Theorem 1 cannot be simplified a priori.

Remark 2. [Are two active states sufficient for approximate capacity characterization?] Consider a line network with one relay. For this network, the schedule that achieves the approximate capacity has only two active states, which activate the links alternatively. Intuitively, one might think that this would extend to arbitrary number of relays. For example, for a network with $N = 3$, can we achieve the approximate capacity by only considering the listen/transmit states $s_1 = 010$ and $s_2 = 101$? The answer to this question is negative as we illustrate through the following example with $N = 3$ and

$$\ell_1 = 2r,\ \ell_2 = 2r,\ \ell_3 = 3r,\ \ell_4 = r, \tag{16}$$

where $r > 0$. By considering only the two aforementioned states, we can achieve a rate of $r\frac{2}{3}$. However, by applying the expression in (1), we get that the HD approximate capacity for this network is $r\frac{3}{4}$. □

Remark 3. [Can we a priori limit our search over a polynomial number of states?] For the FD line network, we can a priori limit our search for the minimum cut over $N+1$ cuts (instead of 2^N). This reduction in the number of cuts is also possible for the HD line network as we proved in Section II-A. This fact raises the question whether we can also a priori reduce the search space for the active states to a polynomial set (instead of 2^N). This is not possible as we state in the theorem below, which is proved in Appendix A. This result might be due to the fact that the capacity expression in (1) depends on the harmonic mean between two consecutive links. Hence, different from FD, changing the order of the point-to-point link capacities, might also change the value of the approximate capacity. □

Theorem 2. *With only the knowledge that N relays are arranged in a line, the cardinality of the smallest search space of states over which a schedule optimal for approximate capacity can be found is $\Omega(2^{N/3})$.*

Remark 4. Theorem 1 has two promising consequences:

1) The HD approximate capacity of the N-relay line network can be computed in $O(N)$ time. This improves on the result in [15], where the approximate capacity can be found in polynomial-time (but not linear in the worst case) by solving a linear program with $O(N)$ variables.

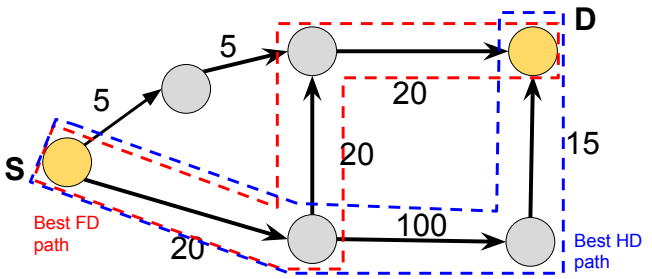

Fig. 3. Example where the best FD and HD paths are different. Edge labels represent the point-to-point link capacities of the edges.

2) The HD approximate capacity in Theorem 1 can be computed in a distributive way as follows. Each relay $i \in [1 : N]$ computes the quantity

$$m_i = \min\left\{\frac{\ell_i\ \ell_{i+1}}{\ell_i + \ell_{i+1}}, m_{i-1}\right\},$$

where $m_0 = \infty$, and sends it to relay $i + 1$. With this, at the end we have $m_N = \mathsf{C}_{\mathcal{R}}$. In other words, for HD approximate capacity computation, it is only required that each relay knows the capacity of its incoming and outgoing links. □

Remark 5. The properties discussed in Remark 4 are the same appealing properties that advocate for the use of the FD capacity in routing protocols. Thus, it is interesting to understand whether routing based on the FD capacities would also give the path with the largest HD approximate capacity or instead routing using the HD approximate capacity expression in (1) would yield different routes. Indeed, it turns out that the FD capacity approach is suboptimal as shown by the example in Fig. 3. By applying the expression in (1), we find that the best HD route (within the blue box in Fig. 3) has an HD approximate capacity of 13.04, which is 30% higher than the HD approximate capacity of the best FD route (within the red box in Fig. 3, with FD capacity of 20 but HD approximate capacity of 10). With best FD (respectively, HD) route we refer to the path that has the largest FD (respectively, HD approximate) capacity. □

The example illustrated in Fig. 3 shows that in general it is suboptimal to find the best HD path by using as optimization metric the FD capacity of the path. In fact, as shown in [25], there exist networks for which routing based on the FD capacities yields a route with HD approximate capacity equal to half that of the best HD route. This observation naturally suggests the question: **Does there exist an efficient (polynomial-time) algorithm that finds the route in a network with the largest HD approximate capacity?** We address this question in the following theorem.

Theorem 3. *For a relay network in the class described in Section II, the problem of finding the best HD path is NP-hard.*

Intuitively, we can attribute the hardness stated in Theorem 3 to the fact that we need to keep track of whether the discovered

paths contain cycles or not, unlike the FD counterpart (a more detailed discussion regarding this aspect can be found in Section IV). This observation suggests that, if the number of cycles in a network is regulated, then we can find the simple path (i.e., a path that contains no cycles) with the largest HD approximate capacity in polynomial-time, as formalized in the lemma below.

Lemma 4. *If the number of cycles in a relay network with N relays (described by the digraph $\mathcal{G}$) is at most polynomial in N (i.e., $O(N^\alpha)$ for some constant α), then we can find the simple path with the largest HD approximate capacity in polynomial-time, particularly in $O((N^\alpha+1)(|\mathcal{E}(\mathcal{G})|\log|\mathcal{E}(\mathcal{G})|+|\mathcal{E}(\mathcal{G})|d))$, where d is the maximum vertex degree in $\mathcal{G}$. This holds even when we do not have an a priori knowledge of the location of the cycles in the network.*

As a network example for which Lemma 4 applies, we can study the layered network where the relays are arranged as M relays per layer over L layers of relays (in total, we have $N = ML$ relays). Every relay can only communicate with the relays in the following layer. It is not difficult to see that for this particular network, the number of cycles in the graph is equal to zero, i.e., $N^\alpha = 0$. In addition, the maximum degree d of a vertex is $O(M)$ and the number of edges in the network is $\Theta(LM^2)$. By substituting these values in the expression in Lemma 4, we get that the complexity of finding a simple path with the largest HD approximate capacity in a layered network is given by

$$\begin{aligned} &O((N^\alpha+1)(|\mathcal{E}(\mathcal{G})|\log|\mathcal{E}(\mathcal{G})|+|\mathcal{E}(\mathcal{G})|d)) \\ &\quad= O(LM^2\log LM^2 + LM^2M) \\ &\quad= O(LM^2\log L + 2LM^2\log M + LM^3) \\ &\quad= O(LM^2\log L + LM^3). \end{aligned}$$

IV. HD Routing is NP-hard

For a network represented by the directed graph $\mathcal{G}$, a path $\mathcal{P} = v_{k_1} - v_{k_2} - \ldots - v_{k_{m+1}}$ of length m in $\mathcal{G}$ is a sequence of vertices $v_{k_i} \in \mathcal{V}(\mathcal{G}), \forall i \in [1:m+1]$. An *$S$-$D$ simple path* in $\mathcal{G}$ is a path for which $v_{k_1} = v_0 = S$ and $v_{k_m+1} = v_{N+1} = D$ and all $m+1$ vertices in $\mathcal{P}$ are distinct, i.e., there are no cycles in $\mathcal{P}$. From Theorem 1, the HD approximate capacity of the S-D simple path $\mathcal{P}$ is given by

$$\mathsf{C}_{\mathcal{P}} = \min_{i\in[2:m]} \left\{ \frac{\ell_{k_{i-1},k_i}\ \ell_{k_i,k_{i+1}}}{\ell_{k_{i-1},k_i} + \ell_{k_i,k_{i+1}}} \right\}. \tag{17}$$

Recall that ℓ_{k_{i-1},k_i} represents the link capacity of the edge from node $v_{k_{i-1}} \in \mathcal{P}$ to node $v_{k_i} \in \mathcal{P}$.

In this section, our goal is to prove Theorem 3, i.e., the problem of finding the best HD route in a network is NP-hard. Towards this end, we start by showing that, if we want to find the path $\mathcal{P}$ with the largest value of $\mathsf{C}_{\mathcal{P}}$ in (17), then we need to restrict our search over *simple* paths.

A. Non-simple Paths are Misleading in HD

Practically, a communication route through a network is expected to be a simple path, i.e, a path that contains no cycles. This is due to the fact that for a non-simple path, e.g., $\mathcal{P}_{\text{cyclic}} = S - v_1 - v_2 - \cdots - v_m - v_2 - D$, we know that – from the degraded nature of the network – the information sent from v_m to v_2 is a noisy version of the information that is already available at v_2 (since v_2 appeared earlier in the path). Thus, for the simple path $\mathcal{P}_{\text{simple}} = S - v_1 - v_2 - D$, we fundamentally have that

$$\mathsf{C}_{\mathcal{P}_{\text{cyclic}}} \le \mathsf{C}_{\mathcal{P}_{\text{simple}}}. \tag{18}$$

This observation is true for both FD and HD paths in the network and therefore the best path (in FD or HD) is naturally a simple path. When routing using the FD capacities (to select the best FD route), this observation turns out to be just a technicality since the expression for the FD capacity already exhibits the fundamental property described in (18). Particularly, we have that $\mathcal{E}(\mathcal{P}_{\text{simple}}) \subseteq \mathcal{E}(\mathcal{P}_{\text{cyclic}})$, which directly implies that

$$\mathsf{C}^{\text{FD}}_{\mathcal{P}_{\text{cyclic}}} = \min_{e_{i,j}\in\mathcal{E}(\mathcal{P}_{\text{cyclic}})} \{\ell_{i,j}\} \le \min_{e_{i,j}\in\mathcal{E}(\mathcal{P}_{\text{simple}})} \{\ell_{i,j}\} = \mathsf{C}^{\text{FD}}_{\mathcal{P}_{\text{simple}}}.$$

Thus, an algorithm that selects a route in FD can end up with either type of paths (simple or cyclic). If the path is cyclic, then we can prune it to get a simple path while ensuring that pruning can only improve the computed capacity.

Differently, for HD routing, it is very important to restrict ourselves to searching over simple paths as the HD approximate capacity expression in (17) only applies to simple paths. Furthermore, applying the expression in (17) to a path with a cycle can actually increase the approximate capacity (in contradiction to the fundamental property in (18)). To illustrate this, consider the network example shown in Fig. 4. From Fig. 4, we now focus on the two paths: the simple path $\mathcal{P}_1 = S - v_1 - D$ and the non-simple path $\mathcal{P}_2 = S - v_1 - v_2 - v_3 - v_1 - D$. Note that $\mathcal{P}_1$ is a simple path and $\mathcal{P}_2$ is a cyclic extension of $\mathcal{P}_1$ by adding the cycle $v_1 - v_2 - v_3 - v_1$. If we apply the expression in (17) on both paths, we get the value equal to 7.5 for $\mathcal{P}_1$ and for $\mathcal{P}_2$ we get 13.05. Thus, if an algorithm is allowed to output non-simple paths, then it would output the path $\mathcal{P}_2$ even though we know fundamentally that $\mathsf{C}_{\mathcal{P}_1} \ge \mathsf{C}_{\mathcal{P}_2}$. This is the first major problem that arises when we allow an algorithm to consider non-simple paths based on the expression in (17). The second problem arises when we observe that $\mathcal{P}_3 = S - v_4 - D$ in Fig. 4 is actually the best HD simple path from S to D. However, since applying (17) for $\mathcal{P}_2$ yields 13.05, which is larger than what

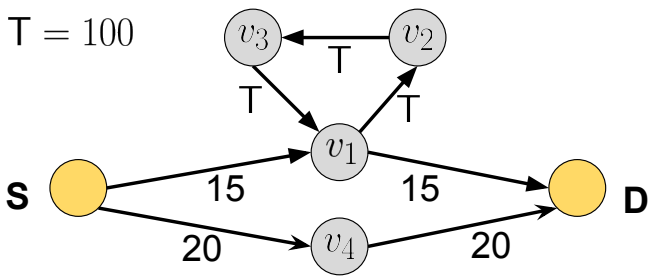

Fig. 4. A network example in which a non-simple path can appear to have a larger HD approximate capacity than its simple subpath.

we get for $\mathcal{P}_3$ (i.e., 10), then the algorithm will output a non-simple path $\mathcal{P}_2$ which when pruned does not yield the best HD path. Thus, an algorithm designed with the goal to find the best HD path needs to be aware of the type of paths that it processes. In other words, we can no longer rely on pruning non-simple paths that an algorithm outputs as these in HD can mislead the algorithm into not selecting the best HD path as illustrated in this example.

As a consequence of the above discussion, in the rest of the section, we focus on the problem of finding the simple (i.e., acyclic) path with the largest HD approximate capacity.

B. Finding the Best HD Simple Path is NP-hard

Our goal in this subsection is to prove that the *search* problem of finding the S-D simple path with the largest HD approximate capacity in a relay network (represented by the digraph $\mathcal{G}$) is NP-hard. Towards proving this, we first show that the related decision problem "HD-Path", which is defined below, is NP-complete.

Definition 2 (HD-Path problem)**.** Given a directed graph $\mathcal{G}$ and a scalar $\mathsf{Z} > 0$, determine whether there exists an S-D simple path in $\mathcal{G}$ with an HD approximate capacity greater than or equal to Z.

Since the decision problem defined above can be reduced in polynomial-time to finding the $S-D$ simple path with the largest HD approximate capacity, then by proving the NP-completeness of the decision problem in Definition 2, we also prove that the search problem is NP-hard.

The HD-Path problem is NP because, given a guess for a path, we can verify in polynomial-time whether it is simple (i.e., no repeated vertices) and whether its HD approximate capacity is greater than or equal to Z by simply evaluating the expression in (17).

To prove the NP-completeness of the HD-Path problem, we now show that the classical 3SAT problem (which is NP-complete) [22] can be reduced in polynomial-time to the HD-Path decision problem in Definition 2. For the 3SAT problem, we are given a boolean expression B in 3-conjunctive normal form,

$$\begin{aligned} B(x_1, x_2, \ldots, x_n) =& (p_{11} \vee p_{12} \vee p_{13}) \wedge (p_{21} \vee p_{22} \vee p_{23}) \\ & \wedge \cdots \wedge (p_{m1} \vee p_{m2} \vee p_{m3}), \end{aligned} \tag{19}$$

where: (i) B is a conjunction of m clauses $\{C_1, C_2, \ldots, C_m\}$, each containing a disjunction of three literals and (ii) a literal p_{ij} is either a boolean variable x_k or its negation $\bar{x}_k$ for some $k \in [1:n]$. The boolean expression B is *satisfiable* if the variables $x_{[1:n]}$ can be assigned boolean values so that B is true. The 3SAT problem answers the question: *Is the given B satisfiable?* We next prove the main result of this section through the following lemma.

Lemma 5. *A polynomial-time reduction exists from the* 3SAT *problem to the* HD-Path *problem.*

Proof. To prove the claim, we create a sequence of graphs based on the boolean statement B given to the 3SAT problem. In each of these graphs, we show that the existence of a satisfying assignment for B is equivalent to a particular feature in the graph. Finally, we construct a network where the feature equivalent to a satisfying assignment of B is the existence of a simple path with HD approximate capacity greater than or equal to Z. In particular, our proof follows four steps of graph constructions, which are explained in detail in what follows.

Running example. To illustrate these four steps we use the following boolean expression as a running example,

$$B = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_5), \tag{20}$$

where, with the notation in (19), the literals are assigned as

$$(p_{11}, p_{12}, p_{13}) = (\bar{x}_1, x_2, x_3), \tag{21a}$$

$$(p_{21}, p_{22}, p_{23}) = (x_4, x_1, \bar{x}_2), \tag{21b}$$

$$(p_{31}, p_{32}, p_{33}) = (\bar{x}_1, x_3, \bar{x}_5). \tag{21c}$$

Step 1. Assume that the boolean expression B is made of m clauses. For each clause $C_i, i \in [1:m]$ in B, construct a gadget digraph $\mathcal{G}_i$ with vertices $\mathcal{V}(\mathcal{G}_i) = \{t_i, v_{i1}, v_{i2}, v_{i3}, r_i\}$ and edges $\mathcal{E}(\mathcal{G}_i) = \bigcup_{j=1}^{3} \{e_{t_i,v_{ij}}, e_{v_{ij},r_i}\}$. Now we connect the gadget graphs $\mathcal{G}_i, i \in [1:m]$, by adding directed edges $e_{r_i,t_{i+1}}, \forall i \in [1:m-1]$. Finally, we introduce a source vertex S and a destination vertex D and the directed edges e_{S,t_1} and $e_{r_m,D}$. We denote this new graph construction by $\mathcal{G}_B$. Note that each vertex v_{ij} in $\mathcal{G}_B$ represents a literal p_{ij} in the boolean expression B. We call a pair of vertices $(v_{ij}, v_{k\ell})$ in $\mathcal{G}_B$, with $i < k$, as *forbidden* if $p_{ij} = \overline{p_{k\ell}}$ in B.

Let $\mathcal{F}$ be the set of all such forbidden non-ordered pairs. Consider an S-D path $\mathcal{P} = S - t_1 - v_{1\ell_1} - r_1 - t_2 - \cdots - v_{m\ell_m} - r_m - D$ in the graph $\mathcal{G}_B$ that contains at most one vertex from any forbidden pair in $\mathcal{F}$. Using the indexes characterizing the path $\mathcal{P}$, if we set the literals $p_{i\ell_i}$ to be true $\forall i \in [1:m]$, then this is a valid assignment (since, by our definition, $\mathcal{P}$ avoids all forbidden pairs in $\mathcal{F}$). Additionally, since we set one literal to be true in each clause, then this assignment satisfies B. Hence the existence of a path $\mathcal{P}$ in $\mathcal{G}_B$ that avoids forbidden pairs implies that B is satisfiable. Similarly, we can show that if B is satisfiable, then we can construct a path that avoids forbidden pairs in $\mathcal{G}_B$ using any assignment that satisfies B.

Running example. The boolean expression in (20) has $m = 3$ clauses. Hence, we construct 3 gadget digraphs that are connected to form $\mathcal{G}_B$ as represented in Fig. 5. Since each vertex $v_{ij}, i \in [1:m],\ j \in [1:3]$, in $\mathcal{G}_B$ represents a literal p_{ij} in the boolean expression in (20) (i.e., $p_{ij} = v_{ij}$) and the literals are assigned as described in (21), then the set of forbidden pairs is given by

$$\mathcal{F} = \{(v_{11}, v_{22}), (v_{12}, v_{23}), (v_{22}, v_{31})\} \tag{22}$$

as also shown in Fig. 5.

Step 2. Next we modify the set of forbidden pairs $\mathcal{F}$ and the graph $\mathcal{G}_B$ such that each vertex appears at most once in $\mathcal{F}$. For each vertex v_{ij} that appears in at least one forbidden pair of $\mathcal{F}$, define $\mathcal{V}_{\mathcal{F}}(v_{ij}) = \{v_{i'j'} \in \mathcal{V}(\mathcal{G}_B) | (v_{ij}, v_{i'j'}) \in \mathcal{F} \text{ or } (v_{i'j'}, v_{ij}) \in \mathcal{F}\}$. Then, for each $\mathcal{V}_{\mathcal{F}}(v_{ij})$, we create $|\mathcal{V}_{\mathcal{F}}(v_{ij})|$ vertices and we label them as $v_{ij,k\ell}, \forall v_{k\ell} \in \mathcal{V}_{\mathcal{F}}(v_{ij})$. We finally replace the vertex v_{ij} in $\mathcal{G}_B$ with a path connecting

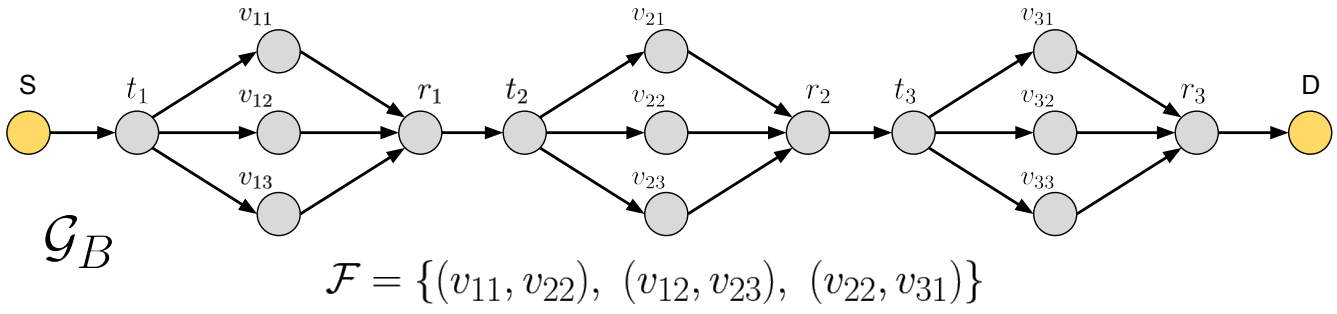

Fig. 5. Graph $\mathcal{G}_B$ and set of forbidden pairs $\mathcal{F}$ for the boolean expression in (20).

the vertices $v_{ij,k\ell}, \forall v_{k\ell} \in \mathcal{V}_{\mathcal{F}}(v_{ij})$. We denote this new graph as $\mathcal{G}_B^{\circ}$. The new set of forbidden pairs $\mathcal{F}^{\circ}$ is defined based on the set $\mathcal{F}$ as $\mathcal{F}^{\circ} = \{(v_{ij,k\ell}, v_{k\ell,ij}) |\ (v_{ij}, v_{k\ell}) \in \mathcal{F}\}$. Note that, for this new set of forbidden pairs, each vertex in $\mathcal{G}_B^{\circ}$ appears in at most one forbidden pair. Let $\mathcal{V}_{\mathcal{F}^{\circ}}$ be the set of vertices that appear in $\mathcal{F}^{\circ}$. Then $\forall v_{ij,kl} \in \mathcal{V}_{\mathcal{F}^{\circ}}$, we replace $v_{ij,kl}$ with a path that consists of three vertices. In particular, for any vertex $v_{ij,k\ell} \in \mathcal{V}_{\mathcal{F}^{\circ}}$, we replace it with a directed path $a_{ij,kl} - v_{ij,kl} - b_{ij,kl}$. We call this new graph $\mathcal{G}_B^{\star}$ and the forbidden pair set $\mathcal{F}^{\star} = \mathcal{F}^{\circ}$. The introduced vertices $a_{ij,k\ell}$ and $b_{ij,k\ell}$ are called *a-type* and *b-type* vertices, respectively.

Similar to our earlier argument for $\mathcal{G}_B$, note that a path in $\mathcal{G}_B^{\star}$ that avoids forbidden pairs in $\mathcal{F}^{\star}$ gives a valid satisfying assignment for the boolean argument B. In the reverse direction, if we have an assignment that satisfies B, then by taking one true literal from each clause C_i, $i \in [1:m]$, we can choose $t_i - r_i$ paths that avoid forbidden pairs. By connecting these paths together, we get an S-D path in $\mathcal{G}_B^{\star}$ that avoids forbidden pairs.

Running example. For our running example, given the set of forbidden pairs $\mathcal{F}$ in (22), we have

$$\begin{aligned}
\mathcal{V}_{\mathcal{F}}(v_{11}) &= \{v_{22}\} \implies v_{11} \leftarrow v_{11,22}, \\
\mathcal{V}_{\mathcal{F}}(v_{22}) &= \{v_{11}, v_{31}\} \implies v_{22} \leftarrow v_{22,11} - v_{22,31}, \\
\mathcal{V}_{\mathcal{F}}(v_{12}) &= \{v_{23}\} \implies v_{12} \leftarrow v_{12,23}, \\
\mathcal{V}_{\mathcal{F}}(v_{23}) &= \{v_{12}\} \implies v_{23} \leftarrow v_{23,12}, \\
\mathcal{V}_{\mathcal{F}}(v_{31}) &= \{v_{22}\} \implies v_{31} \leftarrow v_{31,22},
\end{aligned}$$

where $y \leftarrow \mathcal{Y}$ indicates that in $\mathcal{G}_B^{\circ}$ the vertex y is replaced by the path $\mathcal{Y}$. The set of forbidden pairs $\mathcal{F}^{\circ}$ is then given by

$$\mathcal{F}^{\circ} = \{(v_{11,22}, v_{22,11}), (v_{22,31}, v_{31,22}), (v_{12,23}, v_{23,12})\} \quad (23)$$

and hence $\mathcal{V}_{\mathcal{F}^{\circ}} = \{v_{11,22}, v_{22,11}, v_{22,31}, v_{31,22}, v_{12,23}, v_{23,12}\}$. Given this, we can now construct the graph $\mathcal{G}_B^{\star}$ by replacing any vertex inside $\mathcal{V}_{\mathcal{F}^{\circ}}$ as follows

$$\begin{aligned}
v_{11,22} &\leftarrow a_{11,22} - v_{11,22} - b_{11,22}, \\
v_{22,11} &\leftarrow a_{22,11} - v_{22,11} - b_{22,11}, \\
v_{22,31} &\leftarrow a_{22,31} - v_{22,31} - b_{22,31}, \\
v_{31,22} &\leftarrow a_{31,22} - v_{31,22} - b_{31,22}, \\
v_{12,23} &\leftarrow a_{12,23} - v_{12,23} - b_{12,23}, \\
v_{23,12} &\leftarrow a_{23,12} - v_{23,12} - b_{23,12},
\end{aligned}$$

as shown in Fig. 6 (in order to better visualize the 'evolution' from $\mathcal{G}_B$ to $\mathcal{G}_B^{\star}$, in Fig. 6 we also report again $\mathcal{G}_B$ of Fig. 5). Furthermore, we have $\mathcal{F}^{\circ} = \mathcal{F}^{\star}$, where $\mathcal{F}^{\circ}$ is defined in (23).

Step 3. Our next step is to modify $\mathcal{G}_B^{\star}$ to incorporate $\mathcal{F}^{\star}$ directly into the structure of the graph. For each $(v_{ij,k\ell}, v_{k\ell,ij}) \in \mathcal{F}^{\star}$ introduce a new vertex $f_{ij,k\ell}$ to replace $v_{ij,k\ell}$ and $v_{k\ell,ij}$. All edges that were incident from (to) $v_{ij,k\ell}$ and $v_{k\ell,ij}$ are now incident from (to) $f_{ij,k\ell}$. We call these newly introduced vertices as *f-type* vertices and denote this new graph as $\mathcal{G}_B^{\bullet}$. Note that in $\mathcal{G}_B^{\bullet}$, we now have incident edges from $a_{ij,k\ell}$ and $a_{k\ell,ij}$ to $f_{ij,k\ell}$ and edges incident from $f_{ij,k\ell}$ to vertices $b_{ij,k\ell}$ and $b_{k\ell,ij}$. A path in $\mathcal{G}_B^{\star}$ that avoids forbidden pairs in $\mathcal{F}^{\star}$ gives a path in $\mathcal{G}_B^{\bullet}$ that follows the following rules:

1) **Rule 1:** If any *f-type* vertex is visited, then it is visited at most once;
2) **Rule 2:** If an *f-type* vertex is visited then the preceding *a-type* vertex and the following *b-type* vertex both share the same index (i.e., we do not have $a_{ij,k\ell} - f_{ij,k\ell} - b_{k\ell,ij}$ or $a_{k\ell,ij} - f_{ij,k\ell} - b_{ij,k\ell}$ as a subpath of our path in $\mathcal{G}_B^{\bullet}$).

It is not difficult to see that an $S-D$ path in $\mathcal{G}_B^{\bullet}$ that abides to the two aforementioned rules represents a feasible path that avoids forbidden pairs $\mathcal{F}^{\star}$ in $\mathcal{G}_B^{\star}$. Specifically, this can be seen by treating the subpath $(a_{ij,k\ell} - f_{ij,k\ell} - b_{ij,k\ell})$ in $\mathcal{G}_B^{\bullet}$ as the subpath $(a_{ij,k\ell} - v_{ij,k\ell} - b_{ij,k\ell})$ in $\mathcal{G}_B^{\star}$ and similarly $(a_{k\ell,ij} - f_{ij,k\ell} - b_{k\ell,ij})$ for $(a_{k\ell,ij} - v_{k\ell,ij} - b_{k\ell,ij})$. In other words, the problem of finding a path in $\mathcal{G}_B^{\star}$ that avoids forbidden pairs in $\mathcal{F}^{\star}$ is equivalent to finding a path in $\mathcal{G}_B^{\bullet}$ that satisfies Rule 1 and Rule 2.

Running example. For our running example, the graph $\mathcal{G}_B^{\bullet}$ is shown in Fig. 7. In particular, $\mathcal{G}_B^{\bullet}$ is constructed from $\mathcal{G}_B^{\star}$ in Fig. 6, where each $v_{ij,k\ell} \in \mathcal{V}(\mathcal{G}_B^{\star})$ and $v_{k\ell,ij} \in \mathcal{V}(\mathcal{G}_B^{\star})$ such that $(v_{ij,k\ell}, v_{k\ell,ij}) \in \mathcal{F}^{\star}$, with $\mathcal{F}^{\star}$ being defined in (23), is now replaced by $f_{ij,k\ell}$ in $\mathcal{G}_B^{\bullet}$, which is connected to the other nodes as explained above. In order to better visualize the 'evolution' from $\mathcal{G}_B$ to $\mathcal{G}_B^{\bullet}$, we also report again $\mathcal{G}_B$ of Fig. 5 and $\mathcal{G}_B^{\star}$ of Fig. 6.

Step 4. Our next step is to modify $\mathcal{G}_B^{\bullet}$ by introducing edge capacities. For any edge $e \in \mathcal{E}(\mathcal{G}_B^{\bullet})$ that is not incident from or to an *f-type* vertex, we set the capacity of that edge to be 3Z. For an *f-type* vertex $f_{ij,k\ell}$, let g_1 and h_1 be the link capacities of the edges incident to it from $a_{ij,k\ell}$ and incident from it to $b_{ij,k\ell}$, respectively. Similarly, let g_2 and h_2 be the

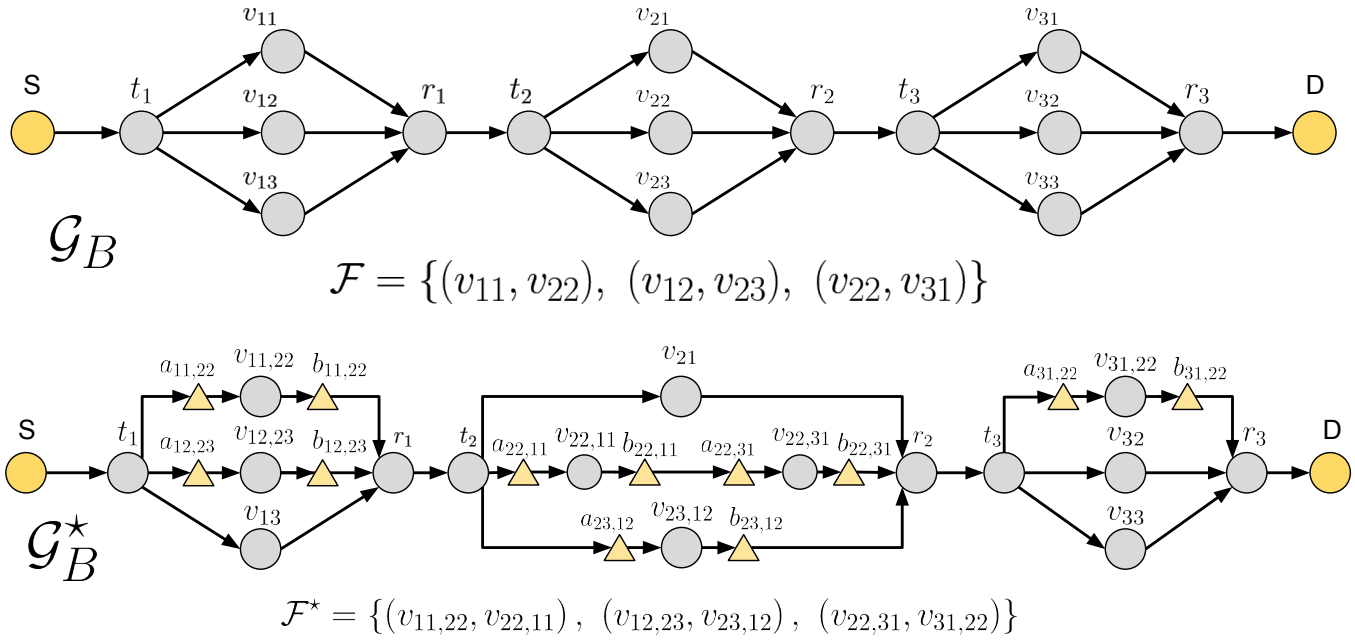

Fig. 6. $\mathcal{G}_B^\star$ and set of forbidden pairs $\mathcal{F}^\star$. The graph $\mathcal{G}_B^\star$ is constructed from $\mathcal{G}_B$.

link capacities of the edges incident from $a_{k\ell,ij}$ and to $b_{k\ell,ij}$, respectively. Then, we set these capacities as

$$g_1 = h_2 = 1.5\mathsf{Z}, \quad g_2 = h_1 = 3\mathsf{Z}.$$

We now need to show that finding a path satisfying Rules 1 and 2 is equivalent to finding a simple path in $\mathcal{G}_B^\bullet$ with HD approximate capacity greater than or equal to Z. It is not difficult to see that a path that follows Rules 1 and 2 is simple and has an HD approximate capacity greater than or equal to Z (by avoiding subpaths $a_{ij,k\ell} - f_{ij,k\ell} - b_{k\ell,ij}$). This is due to the following fact. Consider a path that satisfies Rules 1 and 2. Then, for any two consecutive links in the considered path, at most one of the two links has a capacity of $1.5\mathsf{Z}$ (shown in red in Fig. 7), i.e., at least one link has a capacity of $3\mathsf{Z}$. Thus, a lower bound on the approximate capacity of the considered path is given by $(3\mathsf{Z} \times 1.5\mathsf{Z})/(3\mathsf{Z} + 1.5\mathsf{Z}) = \mathsf{Z}$.

To finally prove the equivalence, we now need to show that a simple path with capacity greater than or equal to Z satisfies Rules 1 and 2. Note that Rule 1 is inherently satisfied since the path is simple (i.e., it visits any vertex at most once). For Rule 2, we next argue that both subpaths are avoided by contradiction.

Assume that the simple path selected contains a subpath of the form $a_{ij,k\ell} - f_{ij,k\ell} - b_{k\ell,ij}$. By our construction of the edge capacities, both the edges $e_{a_{ij,k\ell}, f_{ij,k\ell}}$ and $e_{f_{ij,k\ell}, b_{k\ell,ij}}$ have a capacity equal to $1.5\mathsf{Z}$. This gives us a contradiction since half of the harmonic mean between the capacities of these two consecutive edges is equal to $0.75\mathsf{Z}$. Since the HD approximate capacity of a path is the minimum of half of the harmonic means of its consecutive edges, then the selected path cannot have an HD approximate capacity greater than or equal to Z, which leads to a contraction. Thus, a subpath $a_{ij,k\ell} - f_{ij,k\ell} - b_{k\ell,ij}$ is always avoided. We now need to prove that also the path $a_{k\ell,ij} - f_{ij,k\ell} - b_{ij,k\ell}$ is always avoided. Towards this end, assume that the simple path selected with HD approximate capacity greater than or equal to Z contains (for some i', j', k' and ℓ') a subpath of the form $a_{k'\ell',i'j'} - f_{i'j',k'\ell'} - b_{i'j',k'\ell'}$. Note that, as per our construction in the graph $\mathcal{G}_B^\bullet$, we have that $i' < k'$. Let $i^\star$ be the smallest index i' for which such a subpath exists in our selected path. Since for the subpath in question we have that $i^\star < k'$, then to reach $a_{k'\ell',i^\star j'}$ from S, we have already visited $r_{i^\star}$ earlier in the path. However, to move from $b_{i^\star j',k'\ell'}$ to D (after the subpath in question), we need to pass through $r_{i^\star}$ once more. Clearly, since the path is simple, this leads to a contradiction. Thus, a subpath $a_{k\ell,ij} - f_{ij,k\ell} - b_{ij,k\ell}$ is also always avoided.

This completes the proof that a simple path with capacity greater than or equal to Z satisfies Rule 2. Therefore, finding a path satisfying Rules 1 and 2 is equivalent to finding a simple path in $\mathcal{G}_B^\bullet$ with HD approximate capacity greater than or equal to Z. The second statement is an instance of the HD-Path problem in Definition 2.

Note that in each of the four graph constructions described earlier, we construct one graph from the other using a polynomial number of operations. Thus, this proves by construction that there exists a polynomial reduction from the 3SAT problem to the HD-Path problem. This concludes the proof of Lemma 5 and hence the proof of Theorem 3.

Running example. For our running example, the assignment of the edge capacities is shown in $\mathcal{G}_B^\bullet$ in Fig. 7, where *black* and *blue* edges have a capacity of $3\mathsf{Z}$ and *red* edges have $1.5\mathsf{Z}$. Fig. 7 also shows the evolution of $\mathcal{G}_B$ up to $\mathcal{G}_B^\bullet$.

This concludes the proof of Lemma 5. □

V. Some instances with polynomial-time solutions

In this section, we discuss a special class of networks for which a polynomial-time algorithm exists to find a simple path

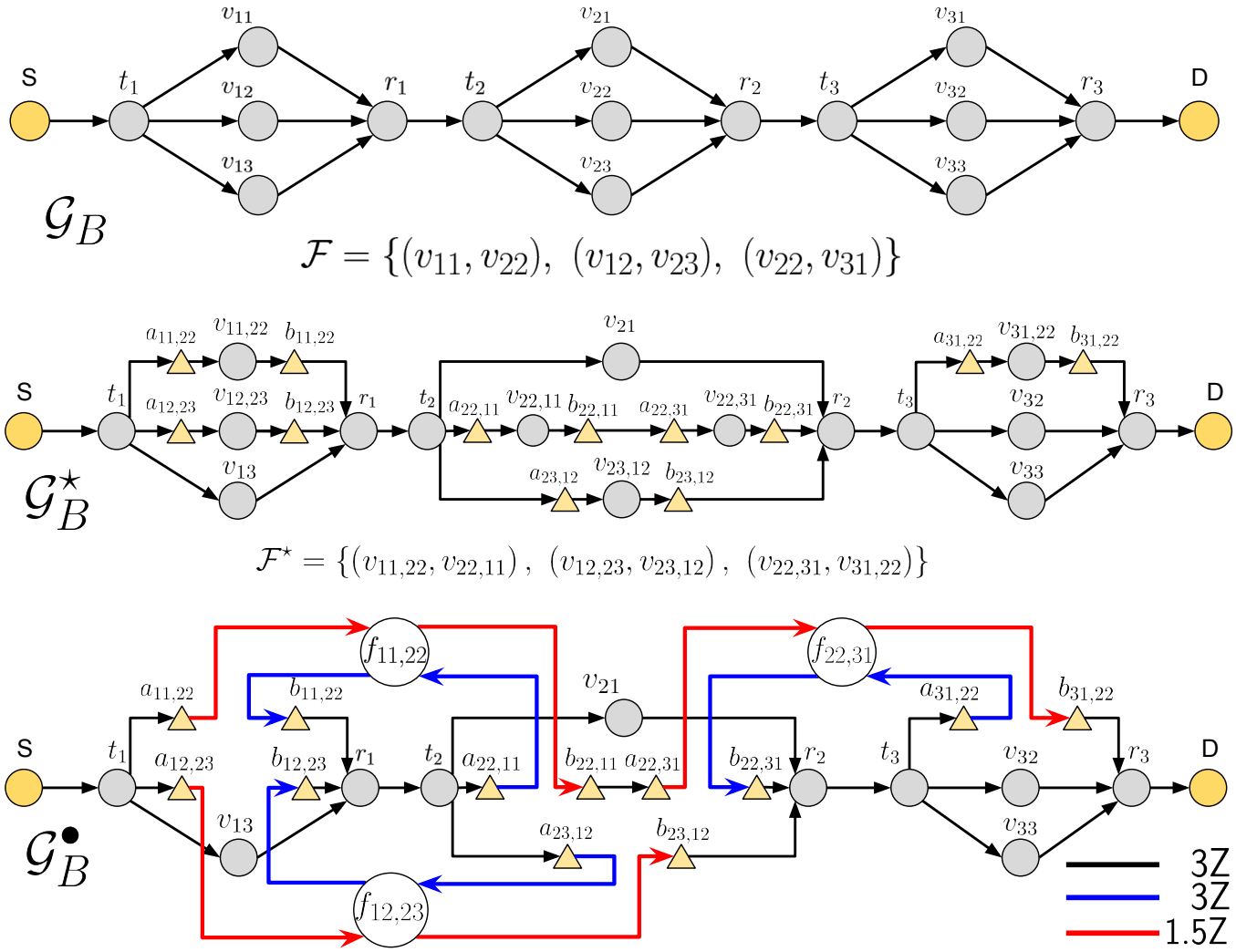

Fig. 7. $\mathcal{G}_B^\bullet$ and the associated edge capacities. The graph $\mathcal{G}_B^\bullet$ is constructed from $\mathcal{G}_B^\star$.

with the largest HD approximate capacity. In particular, we focus on networks where the number of cycles is polynomial, i.e., the number of cycles is at most N^α for some constant $\alpha > 0$, where $N+2$ is the total number of nodes in the network. Our approach is based on relating paths in a network (described by the digraph $\mathcal{G}$) to paths in the line digraph of $\mathcal{G}$ denoted as $\mathcal{L}_\mathcal{G}$. We describe the relation in the next subsection and then present an algorithm that finds the best HD simple path in polynomial-time for the aforementioned class of networks.

A. The Line Digraph Perspective to the Best HD Path Problem

The line digraph of a digraph $\mathcal{G}$ is defined as follows.

Definition 3 (Line digraph $\mathcal{L}_\mathcal{G}$)**.** For a given digraph $\mathcal{G}$, its line digraph $\mathcal{L}_\mathcal{G}$ is a digraph defined by the set of vertices $\mathcal{V}(\mathcal{L}_\mathcal{G})$ and the set of directed edges $\mathcal{E}(\mathcal{L}_\mathcal{G})$. The set $\mathcal{V}(\mathcal{L}_\mathcal{G})$ is defined as $\mathcal{V}(\mathcal{L}_\mathcal{G}) = \{v_{ij} | e_{i,j} \in \mathcal{E}(\mathcal{G})\}$ where $e_{i,j}$ is the directed edge from vertex v_i to vertex v_j. The set of edges $\mathcal{E}(\mathcal{L}_\mathcal{G})$ is defined as $\mathcal{E}(\mathcal{L}_\mathcal{G}) = \{e_{ij,k\ell} | k = j,\ v_{ij}, v_{k\ell} \in \mathcal{V}(\mathcal{L}_\mathcal{G})\}$.

An illustration of a digraph and its associated line digraph is shown in Fig. 8. We can make the following two observations on how simple HD paths are represented in the line digraph.

1) HD paths in $\mathcal{G}$ are equivalent to FD paths in $\mathcal{L}_\mathcal{G}$. Note that a path $\mathcal{P}$ in a network $\mathcal{G}$ can be equivalently defined as the sequence of its adjacent edges (instead of vertices), i.e., we can equivalently write the path $\mathcal{P} = v_{k_1} - v_{k_2} - \cdots - v_{k_m}$ in $\mathcal{G}$ as $\mathcal{P} = e_{k_1,k_2} - e_{k_2,k_3} - \cdots - e_{k_{m-1},k_m}$. Given this and from the definition of the line digraph $\mathcal{L}_\mathcal{G}$, the path $\mathcal{P}$ in $\mathcal{G}$ is equivalent to the path $\mathcal{P}_\mathcal{L} = v_{k_1k_2} - v_{k_2k_3} \cdots - v_{k_{m-1}k_m}$ in $\mathcal{L}_\mathcal{G}$. For each edge $e_{ij,jk} \in \mathcal{E}(\mathcal{L}_\mathcal{G})$, we define the capacity for the edge $e_{ij,jk}$ as

$$c_\mathcal{L}(e_{ij,jk}) = \frac{\ell_{i,j}\ \ell_{j,k}}{\ell_{i,j} + \ell_{j,k}}, \tag{24}$$

where $\ell_{i,j}$ is the point-to-point link capacity of the edge (link) $e_{i,j}$ in $\mathcal{G}$. Thus, we have that the FD capacity of the path $\mathcal{P}_\mathcal{L}$ in $\mathcal{L}_\mathcal{G}$ is given by

$$\begin{aligned} \mathsf{C}^{\mathrm{FD}}_{\mathcal{P}_\mathcal{L}} &= \min_{e_{ij,jk} \in \mathcal{E}(\mathcal{P}_\mathcal{L})} \{c_\mathcal{L}(e_{ij,jk})\} \\ &= \min_{e_{ij,jk} \in \mathcal{E}(\mathcal{P}_\mathcal{L})} \left\{ \frac{\ell_{i,j}\ \ell_{j,k}}{\ell_{i,j} + \ell_{j,k}} \right\} = \mathsf{C}_\mathcal{P}, \end{aligned} \tag{25}$$

where $\mathsf{C}_{\mathcal{P}}$ is defined in (17). From (25) and our previous discussion, we can conclude that, to find the path with the largest HD approximate capacity in the network described by the digraph $\mathcal{G}$, we can first find the path in $\mathcal{L}_{\mathcal{G}}$ that has the largest FD capacity (where the link capacities in $\mathcal{L}_{\mathcal{G}}$ are defined as in (24)) and then map this path in $\mathcal{L}_{\mathcal{G}}$ into its equivalent in $\mathcal{G}$.

2) Simple paths in $\mathcal{G}$ are equivalent to simple chordless paths in $\mathcal{L}_{\mathcal{G}}$. We start by defining *chordal* and *chordless* paths in digraphs.

Definition 4 (Chordal and chordless paths)**.** A path in the digraph $\mathcal{G}'$ is *chordal* if there exists an edge $e \in \mathcal{E}(\mathcal{G}')$ such that its endpoints are two non-consecutive vertices in the path. A path that is not chordal is called *chordless*.

For example, with reference to Fig. 8, the path $S' - S - v_4 - v_2 - v_1 - v_6 - v_3 - D - D'$ is a chordal path in $\mathcal{G}$ since $e_{3,2} \in \mathcal{E}(\mathcal{G})$ and the vertices v_3 and v_2 belong to the path but are non-consecutive. Thus, $e_{3,2}$ is a chord for this path in $\mathcal{G}$. A similar reasoning holds for $e_{S,1}$.

Consider a cyclic path $\mathcal{P}_{\text{cycle}}$ in $\mathcal{G}$. This implies that some vertex $v_k \in \mathcal{P}_{\text{cycle}}$ appears at least twice in the path. Denote with v_{q_1} the node following v_k in its first appearance in $\mathcal{P}_{\text{cycle}}$ and with v_{q_2} the node preceding v_k in its second appearance in the path $\mathcal{P}_{\text{cycle}}$. Then, if we write the line digraph equivalence of $\mathcal{P}_{\text{cycle}}$, we have

$$\mathcal{P}_{\mathcal{L}_{\text{cycle}}} = \cdots - v_{kq_1} - \cdots - v_{q_2k} - \dots \ .$$

From the construction of $\mathcal{E}(\mathcal{L}_{\mathcal{G}})$ in Definition 3, we see that the edge $e_{q_2k,kq_1} \in \mathcal{E}(\mathcal{L}_{\mathcal{G}})$, which implies that $\mathcal{P}_{\mathcal{L}_{\text{cycle}}}$ is chordal. Differently, for a simple path $\mathcal{P}_{\text{simple}}$, any vertex $v_k \in \mathcal{P}_{\text{simple}}$ appears only once. Thus, in the line digraph equivalent path $\mathcal{P}_{\mathcal{L}_{\text{simple}}}$, the index k appears only in two consecutive vertices, which implies that $\mathcal{P}_{\mathcal{L}_{\text{simple}}}$ is chordless. This shows the equivalence described in our observation between simple paths in $\mathcal{G}$ and simple chordless paths in $\mathcal{L}_{\mathcal{G}}$.

Given the two observations above, we can now equivalently describe our HD routing problem on the line digraph as follows: *Can we find the chordless simple path in $\mathcal{L}_{\mathcal{G}}$ that has the largest FD capacity?*

B. An Algorithm on the Line Digraph $\mathcal{L}_{\mathcal{G}}$

The goal of the algorithm described in this section is to find the chordless simple path in $\mathcal{L}_{\mathcal{G}}$ that has the largest FD capacity. The algorithm described here is a modification of the result proposed in [26] for selecting shortest paths while avoiding forbidden subpaths in undirected graphs. The result in [26] needs to apply special care when fixing forbidden subpaths in a graph, due to the general unstructured nature of the forbidden set. In contrast, in our setting we will leverage the structured nature of our forbidden subpaths (chordal paths) and our line digraph $\mathcal{L}_{\mathcal{G}}$ to reduce the number of steps when breaking down a discovered chordal path (presented later in Step 3 of the algorithm). In particular, we make use of the fact that the first chord (corresponding to a chordal subpath) encountered within a selected path in the line digraph represents the smallest cycle encountered along the selected path in the original graph $\mathcal{G}$.

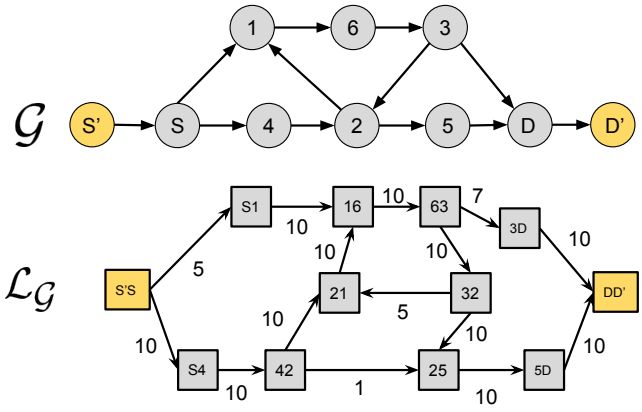

Fig. 8. An example of a digraph $\mathcal{G}$ with its corresponding line digraph $\mathcal{L}_{\mathcal{G}}$. For ease of notation, indexes ij instead of v_{ij} are used and the edge capacities are only shown on $\mathcal{L}_{\mathcal{G}}$.

Thus, by eliminating this subpath, we are sure that the number of remaining chordal paths (present on other paths or larger chordal paths on the same path) has not increased.

To start, we first modify our given network (described by $\mathcal{G}$) so that the source S and the destination D have at most degree one. In particular, we modify the digraph $\mathcal{G}$ by adding two new nodes (namely, S' and D') that are connected only to S and D with edges $e_{S',S}$ and $e_{D,D'}$ (similar to Fig. 8). These two added edges have point-to-point capacities equal to $X \to \infty$. Denote this new digraph by $\mathcal{G}'$ and create the line digraph associated with $\mathcal{G}'$ and denote it by $\mathcal{L}_{\mathcal{G}}^{(0)}$. In $\mathcal{L}_{\mathcal{G}}^{(0)}$, we now consider the node $v_{S'S}$ as our source and the node $v_{DD'}$ as our intended destination.

The algorithm is based on incrementally applying Dijkstra's algorithm [21]. We first try to find the best FD path from $v_{S'S}$ to $v_{DD'}$ in $\mathcal{L}_{\mathcal{G}}^{(i)}$ by running Dijkstra's algorithm. Note that Dijkstra's algorithm returns a spanning tree rooted at $v_{S'S}$ that describes the best FD path from $v_{S'S}$ to each vertex v' in $\mathcal{L}_{\mathcal{G}}^{(i)}$. We denote the tree from our initial run as $\mathcal{T}_0$. From this point, the algorithm iterates (until termination) over four main steps described as follows (starting with $i = 0$).

Step 1. Given the line digraph $\mathcal{L}_{\mathcal{G}}^{(i)}$ and an existing best FD path spanning tree $\mathcal{T}_i$, check whether the path $\mathcal{P}_{\mathcal{L}}^{(i)}$ from $v_{S'S}$ to $v_{DD'}$ defined by $\mathcal{T}_i$ is chordless. If it is chordless, terminate the algorithm since we have found the chordless path from $v_{S'S}$ to $v_{DD'}$ with the largest FD capacity. Otherwise, if it is not chordless, then proceed to Step 2.

Running example. We use the line digraph from Fig. 8 as our $\mathcal{L}_{\mathcal{G}}^{(0)}$. Then, for $i = 0$, we have the spanning tree $\mathcal{T}_0$ (from Dijkstra's algorithm) and the selected path $\mathcal{P}_{\mathcal{L}}^{(0)}$ as shown in Fig. 9. The path $\mathcal{P}_{\mathcal{L}}^{(0)}$ is chordal since $e_{32,21} \in \mathcal{L}_{\mathcal{G}}^{(0)}$ and $e_{42,25} \in \mathcal{L}_{\mathcal{G}}^{(0)}$.

Step 2. Let $\mathcal{C}_{\mathcal{P}}^{(i)}$ be the set of edges in $\mathcal{L}_{\mathcal{G}}^{(i)}$ that are chords for the path $\mathcal{P}_{\mathcal{L}}^{(i)}$ from $v_{S'S}$ to $v_{DD'}$ discussed in the earlier step. Let $\mathcal{C}_{\mathcal{P},\text{first}}^{(i)} \in \mathcal{C}_{\mathcal{P}}^{(i)}$ be the first chord that appears along the path $\mathcal{P}_{\mathcal{L}}^{(i)}$. We denote the endpoints of $\mathcal{C}_{\mathcal{P},\text{first}}^{(i)}$ as $v_{k_1k_2}$

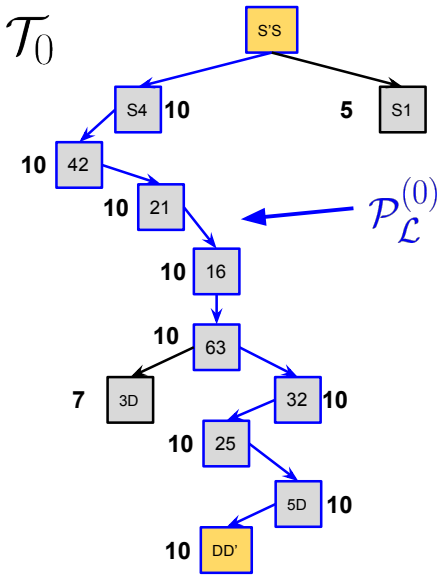

Fig. 9. Tree $\mathcal{T}_0$ for $\mathcal{L}_{\mathcal{G}}^{(0)} = \mathcal{L}_{\mathcal{G}}$ in Fig. 8 (indexes ij instead of v_{ij} are used for ease of notation). Boldface numbers represent the FD capacity with which a node can be reached from SS' using $\mathcal{T}_0$. The highlighted path is the route selected from this tree $\mathcal{T}_0$ from $S'S$ to DD'.

and $v_{k_m k_{m+1}}$, where $v_{k_1 k_2}$ is the vertex that among the two appears earlier in the path $\mathcal{P}_{\mathcal{L}}^{(i)}$ and where m is the length of the subpath $\mathcal{P}_{\text{to-fix}}^{(i)}$ of $\mathcal{P}_{\mathcal{L}}^{(i)}$ connecting the two endpoints, i.e., we now have a path $\mathcal{P}_{\text{to-fix}}^{(i)} = v_{k_1 k_2} - v_{k_2 k_3} - \cdots - v_{k_m k_{m+1}}$. Notice that, with this, we have $k_{m+1} = k_1$.

Running example. For our running example and $i = 0$, we can see from Fig. 8 and Fig. 9 that the set of chords for $\mathcal{P}_{\mathcal{L}}^{(0)}$ is $\mathcal{C}_{\mathcal{P}}^{(0)} = \{e_{32,21}, e_{42,25}\}$. The selected chord $\mathcal{C}_{\mathcal{P},\text{first}}^{(0)}$ is $e_{32,21}$ because its effect on the path concludes earlier than $e_{42,25}$. Hence, we have $\mathcal{P}_{\text{to-fix}}^{(0)} = v_{21} - v_{16} - v_{63} - v_{32}$, which is of length $m = 4$.

Step 3. We now introduce new vertices to the graph $\mathcal{L}_{\mathcal{G}}^{(i)}$ by replicating every intermediate vertex in $\mathcal{P}_{\text{to-fix}}^{(i)}$. In particular, we introduce a replica vertex $v_{k_i' k_{i+1}'}$ for $v_{k_i k_{i+1}}$ where $i \in [2 : m-1]$. We connect these replicas of vertices to each other in the same way their corresponding originals are connected in $\mathcal{P}_{\text{to-fix}}^{(i)}$, i.e., we include the edge $e_{k_i' k_{i+1}', k_{i+1}' k_{i+2}'} \forall i \in [2 : m-1]$ with the same edge capacity as $e_{k_i k_{i+1}, k_{i+1} k_{i+2}}$.

Then, for every $v_{i'j'} \in \mathcal{V}(\mathcal{L}_{\mathcal{G}}^{(i)}) \backslash \mathcal{V}(\mathcal{P}_{\text{to-fix}}^{(i)})$ such that $e_{i'j', k_i k_{i+1}} \in \mathcal{E}(\mathcal{L}_{\mathcal{G}}^{(i)})$, we add an edge that connects $v_{i'j'}$ to the replica vertex of $v_{k_i k_{i+1}}$, i.e., we add the edge $e_{i'j', k_i' k_{i+1}'}$ (with the same edge capacity as $e_{i'j', k_i k_{i+1}}$). In other words, every vertex in $\mathcal{L}_{\mathcal{G}}^{(i)}$ that is not in $\mathcal{P}_{\text{to-fix}}^{(i)}$ and has an edge incident on an intermediate vertex $v_{k_i k_{i+1}}, i \in [2 : m-1]$, of $\mathcal{P}_{\text{to-fix}}^{(i)}$ now has a similar (replicated) edge incident on the replica $v_{k_i' k_{i+1}'}$ of $v_{k_i k_{i+1}}$. Note that at this point: (i) the original vertices in $\mathcal{P}_{\text{to-fix}}^{(i)}$ still form a chordal path in $\mathcal{L}_{\mathcal{G}}^{(i)}$ and (ii) the replica vertices have every possible incident connection their original vertices had except connections to the two endpoint vertices of $\mathcal{P}_{\text{to-fix}}^{(i)}$. We denote the digraph at this point as $\widehat{\mathcal{L}}_{\mathcal{G}}^{(i+1)}$.

Now, our last change is to modify how the two endpoints

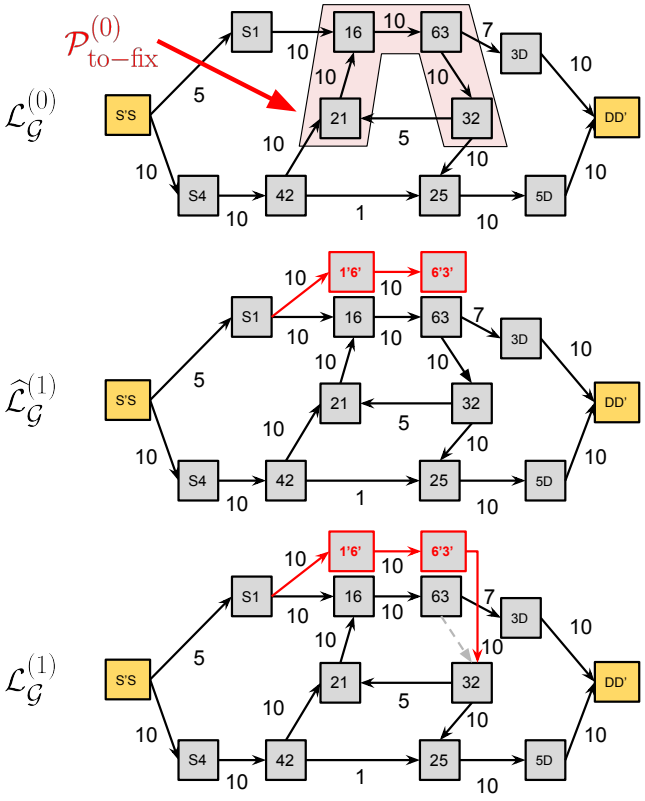

Fig. 10. $\mathcal{L}_{\mathcal{G}}^{(0)}$ from Fig. 8 and the corresponding $\widehat{\mathcal{L}}_{\mathcal{G}}^{(1)}$ and $\mathcal{L}_{\mathcal{G}}^{(1)}$. The replica vertices and the added edges are shown in red while the deleted edges are dashed.

of the path $\mathcal{P}_{\text{to-fix}}^{(i)}$ in $\widehat{\mathcal{L}}_{\mathcal{G}}^{(i+1)}$ connect to the intermediate vertices of the path and their replicas. We do this by adding the edge $e_{k'_{m-1} k'_m, k_m k_{m+1}}$ that connects the last replicated vertex $v_{k'_{m-1} k'_m}$ to the endpoint $v_{k_m k_{m+1}}$ of $\mathcal{P}_{\text{to-fix}}^{(i)}$ and by removing the edge $e_{k_{m-1} k_m, k_m k_{m+1}}$ that connected the original last intermediate vertex to the endpoint. In particular, the new edge $e_{k'_{m-1} k'_m, k_m k_{m+1}}$ has the same capacity as $e_{k_{m-1} k_m, k_m k_{m+1}}$ that was removed. Denote this new digraph as $\mathcal{L}_{\mathcal{G}}^{(i+1)}$. Note that in this new digraph $\mathcal{L}_{\mathcal{G}}^{(i+1)}$, the path $\mathcal{P}_{\text{to-fix}}^{(i)}$ does not exist anymore, while all the other chordless paths have stayed the same. This is due to the fact that we consider the first encountered chord and therefore vertex replication applied in this step either leaves larger chordal subpaths unaffected or eliminates them. This structure is the key step that allowed us to perform a simpler form of vertex replication as compared to [26]. Thus, we have successfully eliminated a cycle (chordal path) that appeared in the digraph before by replicating vertices and deleting edges.

Running example. For our running example and iteration $i = 0$, recall that the chordal path that we would like to fix is given by $\mathcal{P}_{\text{to-fix}}^{(0)} = v_{21} - v_{16} - v_{63} - v_{32}$ (see Fig. 10), where there is a chord due to v_{21} and v_{32}. Only the intermediate vertices of $\mathcal{P}_{\text{to-fix}}^{(0)}$, v_{16} and v_{63} are replicated, while the endpoints v_{21} and v_{32} are unchanged. To generate $\mathcal{L}_{\mathcal{G}}^{(1)}$, we first create $\widehat{\mathcal{L}}_{\mathcal{G}}^{(1)}$ by replicating the intermediate nodes v_{16} and v_{63} (denoted as $v_{1'6'}$ and $v_{6'3'}$) and all incident edges on them that are not

part of $\mathcal{P}^{(0)}_{\text{to-fix}}$. This is shown in Fig. 10. In this case, the only such edge is $e_{S1,16}$ which is replicated by introducing edge $e_{S1,1'6'}$ with the same capacity. To arrive at $\mathcal{L}^{(1)}_{\mathcal{G}}$, we finally remove the last edge in $\mathcal{P}^{(0)}_{\text{to-fix}}$ that connects v_{63} to v_{32} and replace it with an edge connecting $v_{6'3'}$ to v_{32}. In this case, the chordal path $\mathcal{P}^{(0)}_{\text{to-fix}}$ is eliminated (by removing $e_{63,32}$), while all other paths of the type $\cdots - v_{21} - v_{16} - v_{63} - \cdots$ are still available from the remaining part of $\mathcal{P}^{(0)}_{\text{to-fix}}$. Additionally any path that would have used $v_{16} - v_{63} - v_{32}$ (for example $v_{S1} - v_{16} - v_{63} - v_{32}$) is now served by a replica path through the sequence of vertices $v_{S1} - v_{1'6'} - v_{6'3'} - v_{32}$. Thus, we have removed the chordal path $\mathcal{P}^{(0)}_{\text{to-fix}}$ and kept all the other possible paths unchanged or replaced them with a replica. The new generated digraphs $\widehat{\mathcal{L}}^{(1)}_{\mathcal{G}}$ and $\mathcal{L}^{(1)}_{\mathcal{G}}$ are shown in Fig. 10.

Step 4. In the fourth step, our goal is to create the spanning tree $\mathcal{T}_{i+1}$ of the best FD paths associated with the digraph $\mathcal{L}^{(i+1)}_{\mathcal{G}}$. To ensure termination of the algorithm, a condition for this construction is that $\mathcal{T}_{i+1}$ should be made as similar as possible to $\mathcal{T}_i$ [26]. To do so, we run Dijkstra's algorithm to find $\mathcal{T}_{i+1}$ but we start at an intermediate stage in the algorithm, since we already know part of the spanning tree from $\mathcal{T}_i$. In particular, we do the following procedure. Recall our definition of $\mathcal{P}^{(i)}_{\text{to-fix}}$ and its endpoint $v_{k_m k_{m+1}}$ in Step 2. Define $\mathcal{V}^{(i+1)}_{\text{redo}}$ to be the set of vertices for which we need to find a new best FD path. In particular, define $\mathcal{V}^{(i+1)}_{\text{redo}}$ as the union of: (i) the set of all replica vertices introduced in $\mathcal{L}^{(i+1)}_{\mathcal{G}}$, (ii) the set of descendant vertices of $v_{k_m k_{m+1}}$ in $\mathcal{T}_i$, and (iii) the vertex $v_{k_m k_{m+1}}$. For any vertex $v \notin \mathcal{V}^{(i+1)}_{\text{redo}}$, the path connecting $v_{SS'}$ to v in $\mathcal{T}_i$ does not pass through $\mathcal{P}^{(i)}_{\text{to-fix}}$. As a result, we can copy this part of $\mathcal{T}_i$ to $\mathcal{T}_{i+1}$ without loss of generality. Clearly, replica vertices never existed before $\mathcal{L}^{(i+1)}_{\mathcal{G}}$ so there is no known path for them in $\mathcal{T}_i$. Similarly, the path from $v_{S'S}$ to $v_{k_m k_{m+1}}$ (and its descendants) passes through $\mathcal{P}^{(i)}_{\text{to-fix}}$, thus, we need to find a new route for them now that the chordal path has been removed from the graph. Also it is not difficult to see that any $v' \notin \mathcal{V}^{(i+1)}_{\text{redo}}$ will not be a descendant of v, $\forall v \in \mathcal{V}^{(i+1)}_{\text{redo}}$ as this would contradict the need to find a new path for some vertex in $\mathcal{V}^{(i+1)}_{\text{redo}}$.

As per our discussion above, we find the rest of $\mathcal{T}_{i+1}$ by initializing an intermediate point in the Dijkstra's algorithm and continue the execution of the algorithm from there. In particular, we start from the point where $\forall v \notin \mathcal{V}^{(i+1)}_{\text{redo}}$ have been expanded (and thus appear in $\mathcal{T}_{i+1}$ with the same path as in $\mathcal{T}_i$). We denote the intermediate version of $\mathcal{T}_{i+1}$ at this point as $\mathcal{T}'_{i+1}$, which is a pruned version of the tree $\mathcal{T}_i$. Note that, at any iteration of the classical Dijkstra's algorithm, a yet to be expanded vertex v has a best so-far path from $v_{S'S}$ of FD capacity $c'(v)$. This achievable FD capacity at an unexpanded vertex v is based on the maximum capacity achieved by each of the neighbor vertices that have already been expanded and added to the spanning tree $\mathcal{T}'_{i+1}$ as well as the capacities of incident edges from those neighbor vertices to the vertex v. We denote the capacity of a neighbor vertex v' that was already expanded as $\hat{c}^{\mathcal{T}_{i+1}}_{\mathcal{L}}(v')$. We now note that the point from which we are going to start Dijkstra's algorithm is when the set of unexpanded vertices is $\mathcal{V}^{(i+1)}_{\text{redo}}$ and the vertices in $\mathcal{T}'_{i+1}$ form the complement set $\mathcal{V}^{(i+1)^c}_{\text{redo}}$. Thus, for the vertices still unexpanded (i.e., those in $\mathcal{V}^{(i+1)}_{\text{redo}}$), the capacities currently achievable at them at this stage of the algorithm are initialized as

$$\widehat{c}_{\mathcal{L}}(v) = \max_{v' \notin \mathcal{V}^{(i+1)}_{\text{redo}}} \min\left\{c^{\mathcal{T}_i}_{\mathcal{L}}(v),\ c_{\mathcal{L}}(e_{v',v})\right\}.$$

Now that we have the initialization of Dijkstra's algorithm to the state that we want, we run the standard routine of the algorithm to continue expanding the vertices in $\mathcal{V}^{(i+1)}_{\text{redo}}$. When all the vertices have been expanded, we get the final tree $\mathcal{T}_{i+1}$.

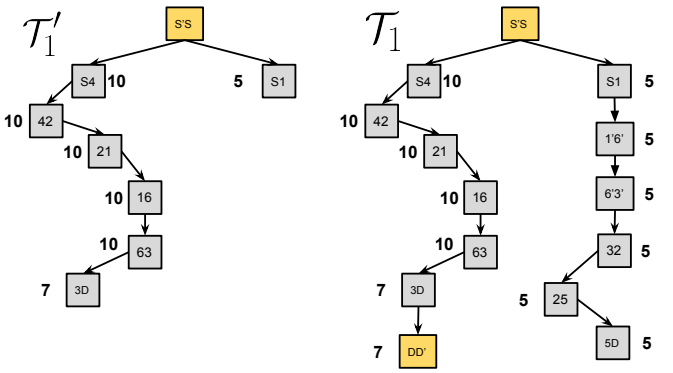

Fig. 11. Spanning trees $\mathcal{T}'_1$ and $\mathcal{T}_1$ for $\mathcal{L}^{(1)}_{\mathcal{G}}$ in Fig. 10. Boldface numbers represent the FD capacity with which a node can be reached from SS' using each tree.

Running example. For our running example and $i = 0$, the tree $\mathcal{T}'_1$ (which is a subset of $\mathcal{T}_0$) and the new generated tree $\mathcal{T}_1$ for $\mathcal{L}^{(1)}_{\mathcal{G}}$ are shown in Fig. 11. It is worth noting that the spanning tree $\mathcal{T}_1$ in Fig. 11 has the path $\mathcal{P}^{(1)}_{\mathcal{L}} = v_{SS'} - v_{S4} - v_{42} - v_{21} - v_{16} - v_{63} - v_{3D} - v_{DD'}$ of capacity $\mathsf{C}^{\text{FD}}_{\mathcal{P}^{(1)}_{\mathcal{L}}} = 7$ that is chordless. Hence the algorithm returns this path and terminates (see Step 1).

It is important to note that, from the replication procedure we do in Step 3, we add a number of replica vertices equal to the length of $\mathcal{P}^{(i)}_{\text{to-fix}}$ minus two (since we do not replicate the endpoints). Moreover, in addition to the replica vertices, only one endpoint of $\mathcal{P}^{(i)}_{\text{to-fix}}$ is a member of $\mathcal{V}^{(i+1)}_{\text{redo}}$ (i.e., the vertex $v_{k_m k_{m+1}}$). As a result $\left|\mathcal{V}^{(i)}_{\text{redo}}\right| \leq |\mathcal{V}(\mathcal{L}_{\mathcal{G}})|, \forall i$. Thus, the size of the network that Dijkstra's algorithm processes in Step 4 does not increase from one iteration to the next. This implies that Step 4 of the algorithm has a complexity that is at most $O(V_{\mathcal{L}_{\mathcal{G}}} \log V_{\mathcal{L}_{\mathcal{G}}} + E_{\mathcal{L}_{\mathcal{G}}})$, where $V_{\mathcal{L}_{\mathcal{G}}} = |\mathcal{V}(\mathcal{L}_{\mathcal{G}})|$ and $E_{\mathcal{L}_{\mathcal{G}}} = |\mathcal{E}(\mathcal{L}_{\mathcal{G}})|$. The time complexity of Steps 1, 2 and 3 is linear in $V_{\mathcal{L}_{\mathcal{G}}}$ and $E_{\mathcal{L}_{\mathcal{G}}}$. Let $K_{\mathcal{G}}$ be the number of cycles in $\mathcal{G}$. From the observation in Section V-A, this is equal to the number of chordal paths in $\mathcal{L}_{\mathcal{G}}$. Since in each iteration over the four steps, we eliminate one chordal path, then for a line graph with $K_{\mathcal{G}}$ chordal paths, we make at most $K_{\mathcal{G}}$ iterations. As a result, the complexity of the described algorithm for finding the simple chordless path with the largest FD capacity in $\mathcal{L}_{\mathcal{G}}$ is $O\left((K_{\mathcal{G}}+1)(V_{\mathcal{L}_{\mathcal{G}}} \log V_{\mathcal{L}_{\mathcal{G}}} + E_{\mathcal{L}_{\mathcal{G}}})\right)$.

Note that the number of vertices in $\mathcal{L}_{\mathcal{G}}$ is equal to the number of edges in $\mathcal{G}$ and the number of edges in $\mathcal{L}_{\mathcal{G}}$ is

upper bounded by the number of edges in $\mathcal{G}$ multiplied by the maximum vertex degree d. Additionally, the complexity of constructing a line digraph $\mathcal{L}_\mathcal{G}$ from a digraph $\mathcal{G}$ is of order $O(|\mathcal{E}(\mathcal{G})|d)$. Thus, the problem of finding the simple path in $\mathcal{G}$ with the largest HD approximate capacity is equivalent to creating the line digraph $\mathcal{L}_\mathcal{G}$ with FD capacities and then finding the chordless path with the largest FD capacity in that line digraph $\mathcal{L}_\mathcal{G}$. The computational complexity of this procedure is $O\left(|\mathcal{E}(\mathcal{G})|d + (K_\mathcal{G}+1)(V_{\mathcal{L}_\mathcal{G}} \log V_{\mathcal{L}_\mathcal{G}} + E_{\mathcal{L}_\mathcal{G}})\right)$ = $O\left((K_\mathcal{G}+1)(|\mathcal{E}(\mathcal{G})| \log |\mathcal{E}(\mathcal{G})| + |\mathcal{E}(\mathcal{G})|d)\right)$. Now if we let $K_\mathcal{G} = O(N^\alpha)$, we get the expression in Lemma 4 which concludes the proof.

VI. Conclusion

In this paper, we studied the problem of characterizing the HD approximate capacity of the N-relay HD line network and investigated the HD routing problem in networks. Towards this end, our first main result was the closed-form characterization of the HD approximate capacity for an N-relay line network (i.e., a path) as a function of the link capacities. We then developed a polynomial-time algorithm for finding a simple schedule (one with at most $N+1$ active states out of the 2^N possible ones) that achieves the HD approximate capacity of the N-relay line network. To the best of our knowledge, this is the first work which provides a closed-form expression for the approximate capacity of an HD relay network with general number of relays and designs an efficient algorithm to find a simple schedule which achieves it.

By leveraging the derived closed-form expression for the HD approximate capacity, we then proved that finding the path from the source to the destination with the largest HD approximate capacity is NP-hard in general. This represents a surprising result and it is fundamentally different from the FD counterpart, since the path with the largest FD capacity can always be discovered in polynomial-time. Finally, we showed that, if the number of cycles inside the network is polynomial in the number of nodes, then a polynomial-time algorithm exists to find the path with the largest HD approximate capacity.

Acknowledgments

We gratefully thank the anonymous reviewer for suggesting the current version of the achievability proof of Theorem 1, which represents a shorter version of the proof that we proposed in [23].

Appendix A
Proof of Theorem 2

We here prove Theorem 2 by proving the following relations in the following subsections.

1) We first prove that the set of fundamental states in an HD line network is equivalent to the set of fundamental maximum cuts in a FD line network.
2) We next show that the problem of finding the set of fundamental maximum cuts for an N-relay FD line network is equivalent to the problem of finding subsets of non-consecutive integers in $[1:N]$.
3) We prove that the number of subsets of non-consecutive integers in $[1{:}N]$ is at least exponential in N.

A. Set of Fundamental Maximum Cuts

In Section II-A we proved that we can compute the approximate capacity $\mathsf{C}_\mathcal{R}$ in (6) by considering only $N+1$ cuts, which are the same that one would need to consider if the network was operating in FD. These $N+1$ cuts are "fundamental", i.e., they do not depend on the values of the point-to-point link capacities. This implies that we can write (6) as the LP

$$\begin{aligned} \mathsf{C}_\mathcal{R} = \quad & \text{maximize} \quad x \\ & \text{subject to} \quad \mathbf{1}_{N+1} x \le \mathbf{A}\lambda \\ & \text{and} \quad \mathbf{1}_{2^N}^T \lambda = 1,\ \lambda \ge \mathbf{0}_{2^N},\ x \ge 0, \end{aligned} \tag{26a}$$

where $\mathbf{A} \in \mathbb{R}^{(N+1)\times 2^N}$ has non-negative entries

$$[\mathbf{A}]_{i,j} = \hat{\ell}_i^{(j)}, \tag{26b}$$

where: (i) $i \in [1:N+1],\ j \in [1:2^N]$; (ii) $\hat{\ell}_i^{(j)}$ is defined in (7). Clearly, the LP in (26) is feasible. The dual of the LP in (26) is given by

$$\begin{aligned} \mathsf{C}_\mathcal{R} = \quad & \text{minimize} \quad y \\ & \text{subject to} \quad \mathbf{1}_{2^N} y \ge \mathbf{A}^T \mathbf{v} \\ & \text{and} \quad \mathbf{1}_{N+1}^T \mathbf{v} \ge 1,\ \mathbf{v} \ge \mathbf{0}_{N+1}, \end{aligned} \tag{27}$$

where $\mathbf{A}$ is defined in (26b). Since the LP in (27) is a minimization and the entries of $\mathbf{A}$ are non-negative, then it is not difficult to see that, for all optimal solutions of (27), we have $\mathbf{1}_{N+1}^T \mathbf{v} = 1$. As a result, an optimal solution of (27) is a solution of

$$\begin{aligned} \mathsf{C}_\mathcal{R} = \quad & \text{minimize} \quad y \\ & \text{subject to} \quad \mathbf{1}_{2^N} y \ge \mathbf{A}^T \mathbf{v} \\ & \text{and} \quad \mathbf{1}_{N+1}^T \mathbf{v} = 1,\ \mathbf{v} \ge \mathbf{0}_{N+1}. \end{aligned} \tag{28}$$

Since in the LP in (27) we are seeking to minimize the objective function, this implies that at least one of the constraints of the type $\mathbf{1}_{2^N} y \ge \mathbf{A}^T \mathbf{v}$ (i.e., the maximum) is satisfied with equality. We can interpret (28) as the problem of finding the least maximum FD cut among a class of line networks $\mathcal{R}_\mathbf{V}$ derived from the original network $\mathcal{R}$, where $\mathbf{V} = \{\mathbf{v} \in \mathbb{R}^{N+1} |\ \mathbf{v} \ge \mathbf{0}, \|\mathbf{v}\|_1 = 1\}$,where v_i is the i-th element of $\mathbf{v}$. For each $\mathbf{v} \in \mathbf{V}$, we define a line network $\mathcal{R}_\mathbf{v} \in \mathcal{R}_\mathbf{V}$, where the point-to-point link capacities are modified by $\mathbf{v}$ as $\ell_i^{(v)} = \ell_i v_i$.

Let $\mathscr{F}_\mathcal{M}$ be the set of fundamental maximum cuts in a FD line network, i.e., the smallest set of cuts over which we need to search for the maximum cut in FD without explicit knowledge of the values of the link capacities or their ordering. Since $\mathscr{F}_\mathcal{M}$ is the set of fundamental maximum cuts, then it contains a maximum cut for any FD line network. As a result, it also contains the least maximum FD cut among the class of line networks $\mathcal{R}_\mathbf{V}$. With this, the rows of $\mathbf{A}^T$ (constraints in (28)) not corresponding to $\mathscr{F}_\mathcal{M}$ are redundant and can be ignored when trying to find an optimal solution in (28). As a consequence of strong duality, the dual multipliers (the states λ_s in (26)) corresponding to the fundamental maximum cuts in $\mathscr{F}_\mathcal{M}$ are sufficient to find a schedule optimal for approximate capacity. We now prove that, without any knowledge of

the link capacities, we need to consider the network states associated to every element of $\mathscr{F}_{\mathcal{M}}$, i.e., considering only the network states corresponding to a subset of $\mathscr{F}_{\mathcal{M}}$ is not sufficient to achieve the approximate capacity. To prove that, it suffices to provide a network example, where for each $\mathcal{A} \in \mathscr{F}_{\mathcal{M}}$ the state $s_{\mathcal{A}^c} = \mathbb{1}_{\mathcal{A}^c}$ is the unique optimal schedule, i.e., $\lambda_{s_{\mathcal{A}^c}} = 1$. For an arbitrary $\mathcal{A} \in \mathscr{F}_{\mathcal{M}}$, define the line network with the link capacities

$$\ell_i = \begin{cases} 1 & \text{if} \quad i \in \mathcal{M}_{\mathcal{A}} \\ X \to \infty & \text{otherwise} \end{cases},$$

where

$$\mathcal{M}_{\mathcal{A}} = \Big\{ i \in [1{:}N{+}1] \Big| i \in \mathcal{A} \cup \{N{+}1\},\ i{-}1 \in \mathcal{A}^c \cup \{0\} \Big\}.$$

From the previous network construction, it is not difficult to see that the unique optimal schedule (one for which $\mathsf{C}_{\mathcal{R}} = \mathsf{C}_{\mathcal{R}}^{\mathrm{FD}}$) is $s_{\mathcal{A}^c} = \mathbb{1}_{\mathcal{A}^c}$, i.e., $\lambda_{s_{\mathcal{A}^c}}{=}1$. Thus, for this particular network construction, the state $s_{\mathcal{A}^c}$ is necessary and hence we cannot further reduce the sufficient set to a subset of $\mathscr{F}_{\mathcal{M}}$, i.e., we need to consider the network states corresponding to every element of $\mathscr{F}_{\mathcal{M}}$.

This result implies that, to find the smallest set of states over which we should search for an optimal schedule for approximate capacity, we should find the set of maximum cuts in FD and then consider their dual multipliers in (26). In what follows, we focus on estimating the cardinality of the set of fundamental maximum cuts $\mathscr{F}_{\mathcal{M}}$ in a FD line network, which, as shown above, gives the cardinality of the smallest search space for an optimal schedule.

B. Finding the Set of Possible Maximum Cuts through an Equivalent Problem

We start by introducing some definitions, which will be used in the rest of this section.

Definition 5. For a set of consecutive integers $[a{:}b]$, we call $\mathcal{H}$ a "punctured" subset of $[a{:}b]$ if $\forall i, j {\in} \mathcal{H}$ with $i \neq j$, we have $|i{-}j|{>}1$, i.e., $\mathcal{H}$ contains non-consecutive integers of $[a{:}b]$.

Definition 6. We call $\mathcal{H}$ a *"primitive punctured"* subset of $[a{:}b]$ if $\mathcal{H}$ is punctured in $[a{:}b]$ and $\forall i \in [a{:}b]\backslash\mathcal{H}$, $\mathcal{H} \cup \{i\}$ is not a punctured set, i.e., $\mathcal{H}$ is not a subset of any other punctured subset of $[a{:}b]$. We denote by $\mathcal{P}(a, b)$ the collection of all primitive punctured subsets of $[a{:}b]$.

We now use the two above definitions to state the following lemma, which is proved in the rest of this section.

Lemma 6. *The problem of finding the set of possible maximum cuts for a FD line network is equivalent to the problem of finding $\mathcal{P}(1, N+1)$, i.e., the collection of primitive punctured subsets of $[1 : N+1]$.*

Proof. We start by defining two problems, namely P_1 and P_2, which are important for the rest of the proof:

$$\mathsf{P}_1: \quad \max_{\mathcal{A} \subseteq [1:N]} g_1(\mathcal{A}) = \sum_{\substack{i \in \mathcal{A} \cup \{N+1\} \\ i-1 \in \mathcal{A}^c \cup \{0\}}} \ell_i, \tag{29a}$$

$$\mathsf{P}_2: \quad \max_{\substack{\mathcal{B} \subseteq [1:N+1] \\ \mathcal{B} \text{ is punctured}}} g_2(\mathcal{B}) = \sum_{i \in \mathcal{B}} \ell_i. \tag{29b}$$

Note that P_1 is the problem of finding the maximum FD cut in an N-relay line network. To relate the solutions of P_1 and P_2, we make use of the following definition.

Definition 7. Given a problem P, we denote with $\mathrm{suf}(\mathsf{P})$ the smallest set of feasible solutions among which an optimal solution can be found for any instance of the problem P.

The proof is organized as follows:

1) **Step 1:** We prove that P_1 and P_2 are equivalent; as a consequence, there exists a function f such that $\mathrm{suf}(\mathsf{P}_1) = f(\mathrm{suf}(\mathsf{P}_2))$.
2) **Step 2:** Next we prove that $\mathrm{suf}(\mathsf{P}_2) \subseteq \mathcal{P}(1, N+1)$, which implies that
$$\mathrm{suf}(\mathsf{P}_1) \subseteq f(\mathcal{P}(1, N+1)).$$
3) **Step 3:** The previous step implies that the set $\mathcal{M}$ of possible maximum cuts is a subset of $f(\mathcal{P}(1, N+1))$. We finally prove that $\mathcal{M} = f(\mathcal{P}(1, N+1))$.

Once proved, these steps imply that we can map the problem of finding the set of possible maximum cuts for a FD line network to the problem of finding $\mathcal{P}(1, N+1)$. We prove these three steps in Appendix B. □

Example. Consider the FD line network with $N = 7$. To find the set of possible maximum cuts, according to Lemma 6, we need to find $\mathcal{P}(1, 8)$, which is given by

$$\mathcal{P}(1,8) = \Big\{ \{1,4,7\}, \{2,4,7\}, \{2,5,7\}, \{2,5,8\}, \{1,3,5,7\}, \{1,3,6,8\}, \{1,4,6,8\}, \{2,4,6,8\}, \{1,3,5,8\} \Big\}.$$

It turns out that we can retrieve the candidate maximum cuts $\mathcal{A}_i$ from $\mathcal{P}(1, 8)$ as follows:

$$\mathcal{A}_i = \mathcal{H}_i \backslash \{8\},\ \mathcal{H}_i \in \mathcal{P}(1,8), \quad \forall i \in [1 : |\mathcal{P}(1,8)|].$$

To conclude the proof of Theorem 2, we need to understand how the size of $\mathcal{P}(1, N+1)$ grows with N, which is the goal of the following subsection.

C. The Size of the Collection of Primitive Punctured Subsets

In this subsection, we prove that the size of the collection of primitive punctured subsets of $[1 : N+1]$ grows exponentially in N. In particular, we prove the following lemma.

Lemma 7. *Let $T(N)$ be the number of primitive punctured subsets of $[1 : N]$. Then, for all $N \geq 4$, we have the following relation,*

$$T(N) = T(N-2) + T(N-3).$$

The proof of the above lemma can be found in Appendix C.

Remark 6. The result in Lemma 7 suggests that $T(N)$ grows exponentially fast. This can be proved by observing the following lower bound on $T(N)$:

$$T(N) = T(N-2) + T(N-3) \geq 2T(N-3), \tag{30}$$

where the inequality is a consequence of the fact that $T(N-2) \geq T(N-3)$. By recursive application of the bound in (30), we have that

$$T(N) \geq 2T(N-3) \geq 2\left(2T(N-6)\right)$$

$$= 4T(N-6) \geq 4\left(2T(N-9)\right) = \cdots .$$

Thus, we have that $T(N) \geq 2^k T(N-3k)$, $\forall k \in [1 : \lfloor N/3 \rfloor]$. By choosing $k = \lfloor N/3 \rfloor - 1$, then for all $N \geq 4$, we have that

$$\begin{aligned} T(N) &\geq 2^{\lfloor \frac{N}{3} \rfloor - 1} T(N - 3\lfloor N/3 \rfloor + 3) \\ &\geq 2^{\frac{N}{3}-2} T(N - 3\lfloor N/3 \rfloor + 3) \\ &\geq 2^{\frac{N}{3}-2} T(3) = \frac{T(3)}{4} 2^{N/3} \geq \frac{T(1)}{4} 2^{N/3}, \end{aligned}$$

where the last two inequalities follow from the fact that $T(\cdot)$ is a non-decreasing function. The bound proved above implies that $T(N) = \Omega(2^{N/3})$. □

Since the number of candidate active states is equal to the number of candidate maximum cuts in FD (see the discussion in Appendix A-A) and this is equal to the number of primitive punctured subsets of $[1 : N+1]$ (see Lemma 6), then the number of candidate active states grows as $\Omega(2^{N/3})$. This concludes the proof of Theorem 2.

Remark 7. Using the recurrence relation in Lemma 7, it is not difficult to prove that $T(N) = \Theta(\beta^N)$ where β is the unique real root of the polynomial $x^3 - x - 1 = 0$, i.e., $x = 1.325$. □

Appendix B
Proof of Lemma 6

We here prove each of the three steps highlighted in the proof of Lemma 6.

Step 1. We first start by proving that any feasible solution for P_1 in (29a) can be transformed into a feasible solution for P_2 in (29b) with the same value for the objective function, i.e., $\forall \mathcal{A} \subseteq [1 : N]$,

$$\exists \text{ punctured } \mathcal{B}_{\mathcal{A}} \in [1 : N+1], \text{ s.t. } g_1(\mathcal{A}) = g_2(\mathcal{B}_{\mathcal{A}}).$$

To show this, for $\mathcal{A} \subseteq [1 : N]$, we simply define $\mathcal{B}_{\mathcal{A}}$ as

$$\mathcal{B}_{\mathcal{A}} = \Big\{ i \in [1{:}N{+}1] \Big| i \in \mathcal{A} \cup \{N{+}1\}, i{-}1 \in \mathcal{A}^c \cup \{0\} \Big\}. \quad (31)$$

It is clear that $\mathcal{B}_{\mathcal{A}}$ is a punctured set as $\forall i \in \mathcal{B}_{\mathcal{A}},\ i-1 \notin \mathcal{B}_{\mathcal{A}}$. Additionally, (31) directly gives us the desired relation as

$$g_1(\mathcal{A}) = \sum_{\substack{i \in \mathcal{A} \cup \{N+1\} \\ i-1 \in \mathcal{A}^c \cup \{0\}}} \ell_i = \sum_{i \in \mathcal{B}_{\mathcal{A}}} \ell_i = g_2(\mathcal{B}_{\mathcal{A}}). \quad (32)$$

What remains to prove now is that any feasible solution $\mathcal{B}$ for P_2 gives a feasible solution $\mathcal{A}_{\mathcal{B}}$ for P_1 and $g_1(\mathcal{A}_{\mathcal{B}}) = g_2(\mathcal{B})$. For a punctured subset $\mathcal{B}$ of $[1 : N+1]$, let

$$\mathcal{A}_{\mathcal{B}} = f_{\mathcal{AB}}(\mathcal{B}) = \underbrace{\Big\{ i \in [1{:}N] \Big| i > \sup(\mathcal{B}) \Big\}}_{\mathcal{A}_{\text{tail}}} \cup \underbrace{\mathcal{B} \backslash \{N{+}1\}}_{\mathcal{A}_{\text{main}}}. \quad (33)$$

It is not difficult to see that, by applying the transformation in (31) on $\mathcal{A}_{\mathcal{B}}$, we get back $\mathcal{B}$, i.e., $\mathcal{B}_{\mathcal{A}_{\mathcal{B}}} = \mathcal{B}$. This is due to the fact that applying (31) removes $\mathcal{A}_{\text{tail}}$ which is composed of a consecutive number of integers while keeping $\mathcal{A}_{\text{main}}$ which, since $\mathcal{B}$ is punctured, is also punctured. Given this, we can directly see from (32) that $g_1(\mathcal{A}_B) = g_2(\mathcal{B}_{\mathcal{A}_{\mathcal{B}}}) = g_2(\mathcal{B})$. This concludes the proof of Step 1.

Step 2. We prove this step by showing that, if there exists an optimal solution $\mathcal{B}^\star$ for P_2 that is not primitive, then there also exists a primitive punctured set $\mathcal{B}'$ such that $g_2(\mathcal{B}^\star) = g_2(\mathcal{B}')$. Since $\mathcal{B}^\star$ is not a primitive punctured set, then there exists another punctured set $\mathcal{B}'$ such that $\mathcal{B}^\star \subset \mathcal{B}'$ and

$$g_2(\mathcal{B}^\star) = \sum_{i \in \mathcal{B}^\star} \ell_i \leq \sum_{i \in \mathcal{B}'} \ell_i = g_2(\mathcal{B}').$$

If we take the largest such $\mathcal{B}'$, we end up with a primitive punctured set. However, by definition (i.e., since $\mathcal{B}^\star$ is an optimal solution) we have that $\forall \mathcal{B}$ punctured, $g_2(\mathcal{B}) \leq g_2(\mathcal{B}^\star)$. This shows that $g_2(\mathcal{B}^\star) = g_2(\mathcal{B}')$ and therefore, $\text{suf}\,(\mathsf{P}_2) \subseteq \mathcal{P}(1, N+1)$. This concludes the proof of Step 2.

Step 3. In the first two steps, we proved that P_1 and P_2 are equivalent and that $\text{suf}\,(\mathsf{P}_2) \subseteq \mathcal{P}(1, N+1)$. This implies that $\text{suf}\,(\mathsf{P}_1) \subseteq f_{\mathcal{AB}}\left(\mathcal{P}(1, N+1)\right)$, where $f_{\mathcal{AB}}(\cdot)$ is defined in (33). We here prove that $\text{suf}\,(\mathsf{P}_1) = f_{\mathcal{AB}}\left(\mathcal{P}(1, N+1)\right)$. Consider an arbitrary set $\mathcal{A} \in f_{\mathcal{AB}}\left(\mathcal{P}(1, N+1)\right)$. To prove that $\mathcal{A} \in \text{suf}\,(\mathsf{P}_1)$, it suffices to provide a network (an instance of P_1) for which $\mathcal{A}$ is the unique maximizer of P_1. Towards this end, for the selected $\mathcal{A}$, we define $\mathcal{B}_{\mathcal{A}}$ as in (31). We know that $\mathcal{B}_{\mathcal{A}}$ is a primitive punctured set and $g_1(\mathcal{A}) = g_2(\mathcal{B}_{\mathcal{A}})$. Now consider the network with link capacities

$$\ell_i = \begin{cases} 1 & \text{if } i \in \mathcal{B}_{\mathcal{A}} \\ 0 & \text{otherwise} \end{cases}.$$

For this network, it is not difficult to see that $g_2(\mathcal{B}) = |\mathcal{B} \cap \mathcal{B}_{\mathcal{A}}|$, for any punctured set $\mathcal{B}$. We now want to show that $\forall \mathcal{A}' \in f_{\mathcal{AB}}\left(\mathcal{P}(1, N+1)\right) \backslash \mathcal{A}$, we have $g_1(\mathcal{A}') < g_1(\mathcal{A})$. Let $\mathcal{B}_{\mathcal{A}'}$ be defined as in (31). Again, from the proof of the previous steps the set $\mathcal{B}_{\mathcal{A}'}$ is primitive punctured and $g_1(\mathcal{A}') = g_2(\mathcal{B}_{\mathcal{A}'})$. Moreover, since $\mathcal{B}_{\mathcal{A}'}$ and $\mathcal{B}_{\mathcal{A}}$ are both primitive we have that $\mathcal{B}_{\mathcal{A}'} \cap \mathcal{B}_{\mathcal{A}} \subset \mathcal{B}_{\mathcal{A}}$. Thus, we obtain

$$\begin{aligned} g_2(\mathcal{B}_{\mathcal{A}'}) = &|\mathcal{B}_{\mathcal{A}'} \cap \mathcal{B}_{\mathcal{A}}| < |\mathcal{B}_{\mathcal{A}}| = g_2(\mathcal{B}_{\mathcal{A}}) \\ &\implies g_1(\mathcal{A}') < g_1(\mathcal{A}). \end{aligned}$$

Since this is true for any arbitrary $\mathcal{A} \in f_{\mathcal{AB}}\left(\mathcal{P}(1, N+1)\right)$, then it is true $\forall \mathcal{A} \in f_{\mathcal{AB}}\left(\mathcal{P}(1, N+1)\right)$. This implies that each element in $f_{\mathcal{AB}}\left(\mathcal{P}(1, N+1)\right)$ is a unique maximum cut for some network construction. Therefore, without any information about the link capacities ℓ_i, we cannot further reduce the set of possible maximum cuts and thus we have $\text{suf}\,(\mathsf{P}_1) = f_{\mathcal{AB}}\left(\mathcal{P}(1, N+1)\right)$. This concludes the proof of Step 3 and hence the proof of Lemma 6.

Appendix C
Proof of Lemma 7

To compute the size of $\mathcal{P}(a, b)$, it is helpful to first prove some properties of $\mathcal{P}(a, b)$ and primitive punctured subsets that will help throughout the proof.

Property 1. *Let $\mathcal{H}$ be a primitive punctured subset of $[a : b]$, then* $\min\{\mathcal{H}\} \leq a + 1$.

Proof. We prove this result by contradiction. Assume that for some primitive punctured set $\mathcal{H}$, we have $\min\{\mathcal{H}\} \geq a + 2$. This implies that $\mathcal{H} \subset [a + 2 : b]$. Let $\hat{\mathcal{H}} = \mathcal{H} \cup \{a\}$. Since $\mathcal{H}$ is a punctured set, then $\hat{\mathcal{H}}$ is also a punctured set because

$\forall i \in \mathcal{H},\ |a-i| > 1$. But since $\mathcal{H} \subset \hat{\mathcal{H}}$, then $\mathcal{H}$ is not a primitive punctured set, which is a contradiction. □

Property 1 implies that, for a primitive punctured subset of $[a:b]$, the minimum element is either a or $a+1$. Therefore, we can write $\mathcal{P}(a,b)$ as

$$\mathcal{P}(a,b) = \mathcal{P}_1(a,b) \ \uplus\ \mathcal{P}_2(a,b),$$

where $\mathcal{P}_1(a,b)$ (respectively, $\mathcal{P}_2(a,b)$) is the collection of primitive punctured sets with minimum element a (respectively, $a+1$). Clearly, $\mathcal{P}_1$ and $\mathcal{P}_2$ are disjoint (we use $\uplus$ to indicate that the union is over disjoint sets). Next, we prove some properties of $\mathcal{P}_1(a,b)$ and $\mathcal{P}_2(a,b)$.

Property 2. $\mathcal{P}_2(a,b) = \mathcal{P}_1(a+1,b)$.

Proof. Let $\mathcal{H}$ be a primitive punctured subset of $[a+1:b]$ that contains the element $a+1$. $\mathcal{H}$ is also a primitive punctured subset of $[a:b]$. This follows since we cannot add $\{a\}$ to $\mathcal{H}$ to get a larger set of non-consecutive elements. Therefore, $\mathcal{H} \in \mathcal{P}_1(a+1,b) \implies \mathcal{H} \in \mathcal{P}_2(a,b)$. The reverse implication is straightforward since, by definition, $\mathcal{P}_2(a,b)$ is a primitive punctured subset which contains the element $a+1$. □

For the next property, we need to define a new operation on the collection of sets. For a collection of sets $\mathcal{Q}$, let $\{i\} \sqcup \mathcal{Q} = \{\{i\} \cup \mathcal{H} \mid \mathcal{H} \in \mathcal{Q}\}$. We then have the following property.

Property 3. $\mathcal{P}_1(a,b) = \{a\} \sqcup \mathcal{P}(a+2,b)$.

Proof. Let $\mathcal{H}$ be a primitive punctured subset of $[a+2:b]$ and define $\hat{\mathcal{H}} = \{a\} \cup \mathcal{H}$. Since $\mathcal{H}$ is a primitive punctured subset of $[a+2:b]$, this means that $\nexists i \in [a+2:b] \backslash \mathcal{H}$ such that $\{i\} \cup \mathcal{H}$ is a punctured sequence of $[a+2:b]$. This implies that $\nexists i \in [a:b] \backslash [\mathcal{H} \cup \{i\}]$ such that $\{i\} \cup \hat{\mathcal{H}}$ is a punctured sequence of $[a:b]$. Therefore $\hat{\mathcal{H}}$ is a primitive punctured sequence of $[a:b]$, i.e., $\hat{\mathcal{H}} \in \mathcal{P}_1(a,b)$. To prove the reverse, consider $\widetilde{\mathcal{H}} \in \mathcal{P}_1(a,b)$. We need to prove that $\hat{\mathcal{H}} = \widetilde{\mathcal{H}} \backslash \{a\}$ is a primitive punctured subset of $[a+2:b]$. Note that the definition of primitive subset of $[a:b]$ implies that $\forall i \in [a+2:b] \backslash \widetilde{\mathcal{H}}$, $\widetilde{\mathcal{H}} \cup \{i\}$ is not a punctured set. Since $a \notin [a+2:b]$, this implies that $\forall i \in [a+2:b] \backslash \hat{\mathcal{H}}$, $\widetilde{\mathcal{H}} \cup \{i\}$ is not a punctured set. Now note that since $\widetilde{\mathcal{H}} \in \mathcal{P}_1(a,b)$ then $a+1 \notin \widetilde{\mathcal{H}}$. Therefore, $\forall i \in [a+2:b]$ removing the element a from $\widetilde{\mathcal{H}} \cup \{i\}$ does not make it a punctured set. We therefore conclude that, $\forall i \in [a+2:b] \backslash \hat{\mathcal{H}}$, $\hat{\mathcal{H}} \cup \{i\}$ is not a punctured set and as a result $\hat{\mathcal{H}} = \widetilde{\mathcal{H}} \backslash \{a\}$ is a primitive punctured subset of $[a+2:b]$. □

We now have all the necessary tools to prove Lemma 7. We obtain

$$\begin{aligned}\mathcal{P}(1,N) &= \mathcal{P}_1(1,N) \uplus \mathcal{P}_2(1,N) \\ &\overset{(a)}{=} \mathcal{P}_1(1,N) \uplus \mathcal{P}_1(2,N) \\ &\overset{(b)}{=} \big[\{1\} \sqcup \mathcal{P}(3,N)\big] \ \uplus\ \big[\{2\} \sqcup \mathcal{P}(4,N)\big],\end{aligned}$$

where the equality in (a) follows from Property 2 and the equality in (b) follows from Property 3. Now note that

$$\begin{aligned}\big|\,[\{i\} \sqcup \mathcal{P}(a,N)]\,\big| &= \big|\mathcal{P}(a,N)\big| = \big|\mathcal{P}(1,N-a+1)\big| \\ &= T(N-a+1),\end{aligned}$$

since the number of sets in each collection remains the same. Therefore, we have

$$\begin{aligned}T(N) &= |\mathcal{P}(1,N)| \\ &= \big|\,[\{1\} \sqcup \mathcal{P}(3,N)]\ \uplus\ [\{2\} \sqcup \mathcal{P}(4,N)]\,\big| \\ &= \big|\,[\{1\} \sqcup \mathcal{P}(3,N)]\,\big| + \big|\,[\{2\} \sqcup \mathcal{P}(4,N)]\,\big| \\ &= T(N-2) + T(N-3).\end{aligned}$$

This concludes the proof of Lemma 7.

REFERENCES

[1] M. Duarte, A. Sabharwal, V. Aggarwal, R. Jana, K. Ramakrishnan, C. W. Rice, and N. Shankaranarayanan, "Design and characterization of a full-duplex multiantenna system for WiFi networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 3, pp. 1160–1177, 2014.

[2] E. Everett, C. Shepard, L. Zhong, and A. Sabharwal, "SoftNull: Many-Antenna Full-Duplex Wireless via Digital Beamforming," *IEEE Transactions on Wireless Communications*, vol. 15, no. 12, pp. 8077–8092, 2016.

[3] Y.-P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, "A primer on 3gpp narrowband internet of things," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 117–123, 2017.

[4] B. Awerbuch, D. Holmer, and H. Rubens, "High throughput route selection in multi-rate ad hoc wireless networks," in *IFIP Working Conference on Wireless On-Demand Network Systems*, 2004, pp. 253–270.

[5] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003, pp. 134–146.

[6] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, 1998, pp. 85–97.

[7] M. R. Aref, "Information flow in relay networks," *Ph.D. thesis, Stanford University*, 1981.

[8] G. Kramer, "Models and theory for relay channels with receive constraints," in *42nd Annual Allerton Conference on Communication, Control, and Computing*, September 2004, pp. 1312–1321.

[9] A. S. Avestimehr, S. N. Diggavi, and D. N. C. Tse, "Wireless Network Information Flow: A Deterministic Approach," *IEEE Transactions on Information Theory*, vol. 57, no. 4, pp. 1872–1905, April 2011.

[10] A. Özgür and S. N. Diggavi, "Approximately Achieving Gaussian Relay Network Capacity With Lattice-Based QMF Codes," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8275–8294, December 2013.

[11] M. Cardone, D. Tuninetti, R. Knopp, and U. Salim, "Gaussian Half-Duplex Relay Networks: Improved Constant Gap and Connections With the Assignment Problem," *IEEE Transactions on Information Theory*, vol. 60, no. 6, pp. 3559–3575, June 2014.

[12] M. Cardone, D. Tuninetti, and R. Knopp, "On the Optimality of Simple Schedules for Networks With Multiple Half-Duplex Relays," *IEEE Transactions on Information Theory*, vol. 62, no. 7, pp. 4120–4134, July 2016.

[13] S. Brahma, C. Fragouli, and A. Özgür, "On the Complexity of Scheduling in Half-Duplex Diamond Networks," *IEEE Transactions on Information Theory*, vol. 62, no. 5, pp. 2557–2572, May 2016.

[14] L. Ong, M. Motani, and S. J. Johnson, "On capacity and optimal scheduling for the half-duplex multiple-relay channel," *IEEE Transactions on Information Theory*, vol. 58, no. 9, pp. 5770–5784, 2012.

[15] R. Etkin, F. Parvaresh, I. Shomorony, and A. Avestimehr, "Computing Half-Duplex Schedules in Gaussian Relay Networks via Min-Cut Approximations," *IEEE Transactions on Information Theory*, vol. 60, no. 11, pp. 7204–7220, November 2014.

[16] T. Lutz, C. Hausl, and R. Kotter, "Bits through deterministic relay cascades with half-duplex constraint," *IEEE transactions on information theory*, vol. 58, no. 1, pp. 369–381, 2012.

[17] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, 1999, p. 90.

[18] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," *RFC 3626, DOI 10.17487/RFC3626*, 2003.

[19] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile computing*. Springer, 1996, pp. 153–181.
[20] M. Pollack, "Letter to the Editor – The Maximum Capacity Through a Network," *Operations Research*, vol. 8, no. 5, pp. 733–736, 1960.
[21] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[22] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.
[23] Y. H. Ezzeldin, M. Cardone, C. Fragouli, and D. Tuninetti, "Efficiently finding simple schedules in Gaussian half-duplex relay line networks," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 471–475.
[24] X. Song and G. Caire, "Queue-Aware Beam Scheduling for Half-Duplex mmWave Relay Networks," *arXiv preprint arXiv:2001.05586*, 2020.
[25] Y. H. Ezzeldin, M. Cardone, C. Fragouli, and D. Tuninetti, "Network Simplification in Half-Duplex: Building on Submodularity," *IEEE Transactions on Information Theory*, vol. 65, no. 10, pp. 6801–6818, 2019.
[26] M. Ahmed and A. Lubiw, "Shortest Paths Avoiding Forbidden Subpaths," in *26th International Symposium on Theoretical Aspects of Computer Science STACS 2009*, 2009, pp. 63–74.

Yahya H. Ezzeldin is a Ph.D. candidate in the Electrical and Computer Engineering Department at the University of California, Los Angeles (UCLA). He received his B.S. and M.S. degrees in Electronics and Communications Engineering from Alexandria University in 2011 and 2014, respectively. His research interests include network information theory and wireless networks focusing on characterizing operational limits in next-generation wireless networks. He worked as a machine learning platform engineer with Intel Corporation in the summer of 2018. He is the recipient of the UCLA University Fellowship in 2014, the Henry Samueli Fellowship in 2016 and the Dissertation Year Fellowship at UCLA in 2019.

Martina Cardone Martina Cardone is currently an Assistant Professor in the Electrical and Computer Engineering department at the University of Minnesota (UMN). Dr. Cardone received her Ph.D. in Electronics and Communications in 2015 from Télécom ParisTech (with work done at Eurecom in Sophia Antipolis, France). From November 2017 to January 2018, she was a post-doctoral associate in the Electrical and Computer Engineering department at UMN. From July 2015 to August 2017, she was a post-doctoral research fellow in the Electrical and Computer Engineering Department at UCLA Henri Samueli School. She is the recipient of the NSF CRII award in 2019, the second prize in the Outstanding Ph.D. award, Télécom ParisTech, Paris, France and the Qualcomm Innovation Fellowship in 2014. Dr. Cardone's main research interests are in network information theory, network coding, and wireless networks with special focus on their capacity, security and privacy aspects.

Christina Fragouli is a Professor in the Electrical and Computer Engineering Department at UCLA and an IEEE fellow. She received the B.S. degree in Electrical Engineering from the National Technical University of Athens, Athens, Greece, and the M.Sc. and Ph.D. degrees in Electrical Engineering from the University of California, Los Angeles. She has worked at the Information Sciences Center, AT&T Labs, Florham Park New Jersey, and the National University of Athens. She also visited Bell Laboratories, Murray Hill, NJ, and DIMACS, Rutgers University. Between 2006–2015 she was an Assistant and Associate Professor in the School of Computer and Communication Sciences, EPFL, Switzerland. She is an IEEE fellow, has served as an Information Theory Society Distinguished Lecturer, and as an Associate Editor for IEEE Communications Letters, for Elsevier Journal on Computer Communication, for IEEE Transactions on Communications, for IEEE Transactions on Information Theory, and for IEEE Transactions on Mobile Communications. She has also served in several IEEE committees, including the Cover Dissertation Award Committee (chair), the IT Magazine Steering Committee (chair), the IEEE Hamming Medal Committee, the IEEE Kobayashi Award Committee, the IEEE Teaching Award committee and the IT Fellows Committee. Her research interests are in algorithms for network information transfer, network security and privacy.

Daniela Tuninetti Daniela Tuninetti (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from ENST/Télécom ParisTech Paris, France, in 2002, with work done at the Eurecom Institute, Sophia Antipolis, France. She is currently a Professor with the Department of Electrical and Computer Engineering, at the University of Illinois at Chicago (UIC), where she joined in 2005. She was a Post-Doctoral Research Associate with the School of Communication and Computer Science, (EPFL), Swiss Federal Institute of Technology, Lausanne, Switzerland, from 2002 to 2004. Her research interests include ultimate performance limits of wireless interference networks (with a special emphasis on cognition and user cooperation), coexistence between radars and communication systems, multirelay networks, content-type coding, cache-aided systems, and distributed private coded computing. She is currently a Distinguished Lecturer of the Information Theory Society. She received the Best Paper Award from the European Wireless Conference in 2002 and the NSF CAREER Award in 2007. She was named a University of Illinois Scholar in 2015. She was the Editor-in-Chief of the IEEE INFORMATION THEORY SOCIETY NEWSLETTER from 2006 to 2008, an Editor of IEEE COMMUNICATION LETTERS from 2006 to 2009, of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2011 to 2014, and of IEEE TRANSACTIONS ON INFORMATION THEORY from 2014 to 2017.