

Polynomial-time Capacity Calculation and Scheduling for Half-Duplex 1-2-1 Networks

Yahya H. Ezzeldin[†], Martina Cardone^{*}, Christina Fragouli[†], Giuseppe Caire^{*}

[†] UCLA, Los Angeles, CA 90095, USA, Email: {yahya.ezzeldin, christina.fragouli}@ucla.edu

^{*} University of Minnesota, Minneapolis, MN 55404, USA, Email: cardo089@umn.edu

^{*} Technische Universität Berlin, Berlin, Germany, Email: caire@tu-berlin.de

Abstract—This paper studies the 1-2-1 half-duplex network model, where two half-duplex nodes can communicate only if they point “beams” at each other; otherwise, no signal can be exchanged or interference can be generated. The main result of this paper is the design of two polynomial-time algorithms that: (i) compute the approximate capacity of the 1-2-1 half-duplex network and, (ii) find the network schedule optimal for the approximate capacity. The paper starts by expressing the approximate capacity as a linear program with an exponential number of constraints. A core technical component consists of building a polynomial-time separation oracle for this linear program, by using algorithmic tools such as perfect matching polytopes and Gomory-Hu trees.

I. INTRODUCTION

Millimeter wave (mmWave) communication is expected to play an integral role in the Fifth Generation (5G) cellular technology, and to be used in a range of services, such as ultra-high resolution video streaming, vehicle-to-vehicle communication and intelligent buildings broadband coverage [1]. However, although a wide spectrum of promising applications has emerged fast, the focus of theoretical work has been mostly limited to single-hop systems. In this paper, we contribute to the fundamental understanding of networks of mmWave nodes building on our previous results in [2].

In [2], we recently introduced Gaussian 1-2-1 networks, a model that abstracts the directivity of mmWave beamforming. Using this model, we studied the Shannon capacity for arbitrary network topologies that consist of Full-Duplex (FD) mmWave nodes, that is, nodes that can receive and transmit at the same time using two highly directive beams. In particular, in [2] we showed that the capacity of a Gaussian 1-2-1 network with FD mmWave nodes can be approximated to within a universal constant gap¹ in polynomial-time in the network size. We term such approximation as *approximate capacity* in the remainder of the paper.

The focus of this paper is on Gaussian 1-2-1 networks with Half-Duplex (HD) nodes, that is, at any time instance, an HD node can either receive or transmit with a highly directive beam, but not both simultaneously. Our main results are as follows. We design a polynomial-time algorithm that enables to calculate the approximate capacity of a Gaussian HD 1-2-1 network with arbitrary topology. Additionally, we design an algorithm that computes the optimal schedule to achieve this *approximate capacity* in polynomial-time. This is a surprising result: the approximate capacity of HD

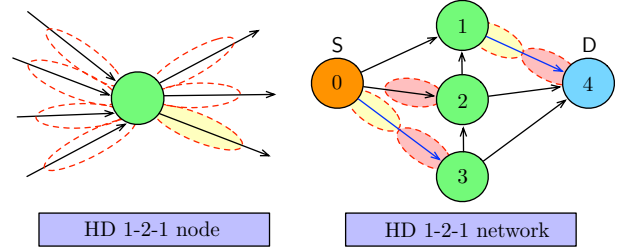


Fig. 1: Model of a 1-2-1 node (left) and an example of a 1-2-1 network with $N = 3$ relays (right).

networks is notoriously hard to study. For a network with N relays, since each node can either transmit or receive, there exist 2^N (an exponential number of) possible states. Thus, to calculate the approximate capacity, we may need to examine the fraction of time each of these states needs to be used. For instance, for the traditional Gaussian wireless network, although for FD networks the approximate capacity can be computed in polynomial-time in the network size [3], in HD such result is known to hold only for few special cases, such as line networks [4] and specific classes of layered networks [5]. Furthermore, although there have been several works that characterize the complexity of the optimal schedule for Gaussian HD wireless networks [6], [7], the problem of efficiently finding the schedule optimal for the approximate capacity for any general number of relays N has only been solved for Gaussian line networks [4]. In this work, we show that the approximate capacity and an optimal schedule for Gaussian HD 1-2-1 networks can be *always* computed in polynomial-time in the network size, independently of the network topology.

Our result for HD 1-2-1 networks parallels our recent result for FD networks proved in [2]. However, the approach and tools we use here are different. In particular, we first show that the approximate capacity can be calculated as the solution of a linear program (LP) that has an exponential number of constraints. Then, we reduce our problem to building a polynomial-time separation oracle for this LP, by using graph-theoretic tools such as perfect matching polytopes [8] and Gomory-Hu trees [9].

Paper Organization. Section II describes the N -relay Gaussian HD 1-2-1 network and presents some known capacity results for them. Section III presents our main results, which are proved in Section IV and Section V.

¹Constant gap refers to a quantity that is independent of the channel coefficients and operating SNR, and solely depends on the number of nodes.

II. SYSTEM MODEL AND KNOWN RESULTS

With $[n_1 : n_2]$ we denote the set of integers from n_1 to $n_2 \geq n_1$; \emptyset is the empty set; $\mathbf{1}_P$ is the indicator function; $\mathbf{0}^N$ is the all-zero vector of length N ; $|S|$ is the absolute value of S when S is a scalar, and the cardinality when S is a set.

1-2-1 Gaussian HD network. We consider a network denoted by \mathcal{N} where N relays assist the communication between a source node (node 0) and a destination node (node $N+1$). We assume the following network model, which is shown in Fig. 1. At any time instant, this network has the two following features: (i) each node can direct/beamform its transmissions towards at most another node, and (ii) each node can point its receiving beam towards – and hence receive transmissions from – at most one other node. Moreover, the N relays operate in HD mode, that is, at any particular time, each relay can be either transmitting to or receiving from at most one node.

In particular, we can mathematically model the features explained above, by introducing two discrete state random variables $S_{i,t}$ and $S_{i,r}$, for each node $i \in [0 : N+1]$. The variable $S_{i,t}$ (respectively, $S_{i,r}$) indicates the node towards which node i is pointing its transmitting (respectively, receiving) beam. With this, we have that

$$S_{i,t} \subseteq [1 : N+1] \setminus \{i\}, \quad |S_{i,t}| \leq 1, \quad (1a)$$

$$S_{i,r} \subseteq [0 : N] \setminus \{i\}, \quad |S_{i,r}| \leq 1, \quad (1b)$$

$$|S_{i,t}| + |S_{i,r}| \leq 1, \quad (1c)$$

where $S_{0,r} = S_{N+1,t} = \emptyset$ since the source always transmits and the destination always receives, and where the constraint in (1c) follows since the relays operate in HD, i.e., for relay $i \in [1 : N]$, if $S_{i,t} \neq \emptyset$, then $S_{i,r} = \emptyset$, and vice versa.

Thus, $\forall j \in [1:N+1]$, we can write the input/output relationship for the Gaussian HD 1-2-1 network as

$$Y_j = \begin{cases} h_{jS_{j,r}} \bar{X}_{S_{j,r}}(j) + Z_j & \text{if } |S_{j,r}| = 1, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where: (i) $Y_j \in \mathbb{C}$ represents the channel output at node j ; (ii) $h_{ji} \in \mathbb{C}$ is the channel coefficient from node i to node j ; the channel coefficients are assumed to remain constant for the entire transmission duration and hence they are known by all nodes in the network; (iii) Z_j is the additive white Gaussian noise at the j -th node; noises across nodes in the network are assumed to be independent and identically distributed as $\mathcal{CN}(0, 1)$; (iv) $\bar{X}_i \in \mathbb{C}^{N+1}$ has elements $\bar{X}_i(k)$ defined as

$$\bar{X}_i(k) = X_i \mathbf{1}_{\{k \in S_{i,t}\}},$$

where $S_{i,t}$ is defined in (1), and where $X_i \in \mathbb{C}$ denotes the channel input at node i ; the channel inputs are subject to an individual power constraint, i.e., $\mathbb{E}[|X_i|^2] \leq P, \forall i \in [0 : N]$; note that, when node i is not transmitting, i.e., $S_{i,t} = \emptyset$, then $\bar{X}_i = \mathbf{0}^{N+1}$; (v) $S_{j,r}$ is defined in (1).

Capacity results on 1-2-1 networks. The Shannon capacity of the Gaussian HD 1-2-1 network described in (2) is not known. However, recently we have shown in [2, Theorem 1]

that the capacity C of the Gaussian HD (as well as FD) 1-2-1 network can be approximated by $C_{\text{cs,iid}}$ as follows²,

$$C_{\text{cs,iid}} \leq C \leq C_{\text{cs,iid}} + \text{GAP}, \quad (3a)$$

$$C_{\text{cs,iid}} = \max_{\substack{\lambda_s: \lambda_s \geq 0 \\ \sum_s \lambda_s = 1}} \min_{\Omega \subseteq [1:N] \cup \{0\}} \sum_{\substack{(i,j): i \in \Omega, \\ j \in \Omega^c}} \underbrace{\left(\sum_{\substack{s: \\ j \in S_{i,t}, \\ i \in S_{j,r}}} \lambda_s \right)}_{A_{ji}^{(s)}} \ell_{j,i}, \quad (3b)$$

$$\ell_{j,i} = \log \left(1 + P |h_{ji}|^2 \right), \quad (3c)$$

$$\text{GAP} = O(N \log N), \quad (3d)$$

where: (i) Ω enumerates all possible cuts in the graph representing the network, such that the source belongs to Ω ; (ii) $\Omega^c = [0 : N+1] \setminus \Omega$; (iii) s enumerates all possible network states of the 1-2-1 network in HD (or FD), where each network state corresponds to specific values for the variables in (1) for each HD node; (iv) λ_s , i.e., the optimization variable, is the fraction of time for which state s is active; we refer to a schedule as the collection of λ_s for all feasible states, such that they sum up to at most 1; (v) $s_{i,t}$ and $s_{i,r}$ denote the transmitting and receiving states for node i in the network state s (defined in (1) for HD operation). In other words, for Gaussian HD 1-2-1 networks, $C_{\text{cs,iid}}$ in (3) is the approximate capacity of the Shannon capacity C .

The expression in (3b) can be explained as follows. Given a fixed schedule $\{\lambda_s\}$, for each point-to-point link ($i \rightarrow j$) in the network, we sum together the activation times λ_s of all the states s that activate this link (represented by A_{ji} in (3b)). Then, we weight/multiply the link capacity $\ell_{j,i}$ by this effective activation time A_{ji} . For this new network with weighted link capacities $\ell_{j,i}^{(s)}$ as shown in (3b), we calculate the graph-theoretic min-cut. Finally, we maximize this min-cut over all possible feasible schedules of the 1-2-1 network.

III. MAIN RESULTS

In this section, we show how the expression of the approximate capacity $C_{\text{cs,iid}}$ in (3b) for Gaussian HD 1-2-1 networks can be efficiently evaluated and how an optimal schedule $\{\lambda_s\}$ for (3b) can be found in polynomial-time in the number of network relays N . In particular, our main result is summarized by the following theorem.

Theorem 1. *For the N -relay Gaussian HD 1-2-1 network:*

- (a) *The approximate capacity $C_{\text{cs,iid}}$ can be found in polynomial time in N ;*
- (b) *An optimal schedule for the approximate capacity $C_{\text{cs,iid}}$ can be found in polynomial-time in N .*

To the best of our knowledge, Gaussian HD 1-2-1 networks represent the first class of HD relay networks for which the approximate capacity and schedule can be computed

²The expression in (3) holds for both HD and FD with different GAP and different feasibility sets of the schedules $\{\lambda_s\}$.

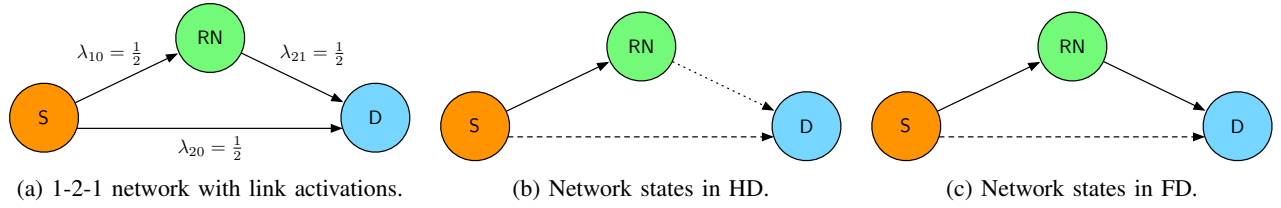


Fig. 2: Gaussian 1-2-1 network examples and network states in HD and FD.

efficiently independently of the network topology. In what follows, we focus on the proof of Theorem 1.

Theorem 1 Part (a). The proof of the first part of Theorem 1 is a direct consequence of two results that we present and discuss in what follows. Our first result shows that calculating $C_{cs,iid}$ for Gaussian HD 1-2-1 networks is equivalent to solving an LP, where state activation times are replaced by link activation times. In particular, we have the following theorem for which the proof is delegated to Appendix A.

Theorem 2. For any N -relay Gaussian HD 1-2-1 network, we have that

$$\begin{aligned}
 \text{P1 : } C_{cs,iid} &= \max \sum_{j=1}^{N+1} F_{j,0} \\
 (1a) \quad &0 \leq F_{j,i} \leq \lambda_{ji} \ell_{j,i}, \quad \forall (i,j) \in [0:N] \times [1:N+1], \\
 (1b) \quad &\sum_{j \in [1:N+1] \setminus \{i\}} F_{j,i} = \sum_{k \in [0:N] \setminus \{i\}} F_{i,k}, \quad i \in [1:N], \\
 (1c) \quad &\hat{\lambda}_{ij} = \lambda_{ij} + \lambda_{ji}, \quad i \in [0:N], \quad j \in [i+1:N+1], \\
 (1d) \quad &\lambda_{ij} \geq 0 \quad (i,j) \in [0:N] \times [1:N+1], \\
 (1e) \quad &\sum_{\substack{(i,j): i=v \text{ or } j=v, \\ i < j}} \hat{\lambda}_{ij} \leq 1, \quad \forall v \in [0:N+1], \\
 (1f) \quad &\sum_{\substack{i \in S, j \in S, \\ i < j}} \hat{\lambda}_{ij} \leq \frac{|S|-1}{2}, \quad \forall S \subseteq [0:N+1], \quad |S| \text{ odd}, \\
 (1g) \quad &\hat{\lambda}_{ij} \geq 0 \quad i \in [0:N], \quad j \in [i+1:N+1],
 \end{aligned}$$

where $F_{j,i}$ represents the data flow through the link of capacity $\ell_{j,i}$ and λ_{ji} represents the fraction of time in which the link is active.

Note that P1 is very similar to the LP representation of the max-flow problem, where the edge capacities are given by $\lambda_{ji} \ell_{j,i}$. The key difference is that λ_{ji} is now a variable and is subject to the feasibility constraints in (1c)–(1f) that stem from the nature of the scheduling in HD 1-2-1 networks. Note that a similar LP as in Theorem 2 was obtained in [2] for the FD case. However, there is a fundamental difference in HD, which is captured by the constraints in (1f) that are not needed in FD. To illustrate the need of the constraint in (1f) in P1, consider the network in Fig. 2a. Assume that each of the three links in the network is active for a fraction of time equal to $1/2$ (as shown in Fig. 2a). Clearly, these link activation times satisfy the constraint in (1c)–(1e) with $\hat{\lambda}_{01} = \hat{\lambda}_{02} = \hat{\lambda}_{12} = \frac{1}{2}$. However, we note that these link activation times do not satisfy the constraint in (1f) since,

by considering $S = \{0,1,2\}$, we have

$$\sum_{i \in S, j \in S, i < j} \hat{\lambda}_{ij} = \hat{\lambda}_{01} + \hat{\lambda}_{02} + \hat{\lambda}_{12} = \frac{3}{2} > \frac{|S|-1}{2} = 1.$$

Thus, if the constraint in (1f) was not there, then one would conclude that the link activation times illustrated in Fig. 2a are feasible. However, we now show that this is not the case, which highlights the need of the constraint in (1f). For the Gaussian 1-2-1 network in Fig. 2a, when the relay operates in HD, there are three possible useful states of the network, in each of which exactly one link is active. These three states are depicted with different line styles (i.e., solid, dashed, dotted) in Fig. 2b. Note that the links $S \rightarrow RN$ and $RN \rightarrow D$ cannot be active simultaneously because of the HD constraint at the relay. Additionally, the links $S \rightarrow RN$ and $S \rightarrow D$ cannot be activated simultaneously since the source has only a single transmitting beam. A similar argument also holds for the links $RN \rightarrow D$ and $S \rightarrow D$. Thus, for this network we have a one-to-one mapping between λ_s in (3b) and λ_{ji} in P1, with $\lambda_s = \lambda_{ji}$ if state s activates the link of capacity $\ell_{j,i}$. Hence, if we use the values from Fig. 2a, we would obtain $\sum_s \lambda_s = 3/2 > 1$ which clearly does not satisfy the constraint in (3b). We therefore conclude that the link activation times illustrated in Fig. 2a, which are feasible if the relay operates in FD (see Fig 2c), are not feasible when the relay operates in HD. This simple example shows why the constraint in (1f) in P1 is needed for HD Gaussian 1-2-1 networks.

We note that the LP P1 has a number of variables that is polynomial in N (two per each edge in the network) compared to the number of variables in the maximization in (3b), which instead is exponential in N (one per each state in the network). However, we also note that P1 now has an exponential number of constraints of the type (1f). Thus, it follows that algorithms such as the simplex method and the interior point method can not solve P1 in polynomial-time in N . However, as we show next, the ellipsoid method [10] can indeed solve P1 in polynomial-time in N . The key step of the ellipsoid method, that incorporates the constraints of an LP, relies on the existence of an oracle which, given the problem and a point in space, can decide in polynomial-time whether the point is feasible or not and, if not, it returns one constraint of the linear program that is violated by that point. This is referred to as a *polynomial-time separation oracle*. Our next result focuses on showing that a polynomial-time separation oracle for P1 exists such that, given the graph representing the N -relay Gaussian HD 1-2-1 network and a point y in the space of P1, it can verify in polynomial-time in N if y is feasible in P1 and, if not feasible, it returns one of

the constraints that is violated. In other words, if one of the constraints is violated, then the oracle returns a hyperplane that separates the point y from the feasible polytope in P1. This result is formalized in the theorem below for which the proof is given in Section IV.

Theorem 3. *A polynomial-time separation oracle exists that, provided with a weighted graph with $N+2$ nodes and a point y in the space of P1, can verify in polynomial-time in N if y is feasible in P1, and if not feasible it returns one of the constraints in P1 that is violated.*

Theorem 2, Theorem 3 and the existence of the ellipsoid method [10] directly imply the result in part (a) of Theorem 1.

Theorem 1 Part (b). The proof of the second part of Theorem 1 makes use of Theorem 3 and an algorithmic version of Caratheodory's theorem to find a feasible schedule λ_s for the approximate capacity in (3b), such that each link is activated for the amount given by the solution of P1. Theorem 1 Part (b) is proved in Section V.

IV. PROOF OF THEOREM 3

In this section, we prove Theorem 3, namely we show the existence of a polynomial time separation oracle that, provided with a weighted graph with $N+2$ nodes – representing our Gaussian HD 1-2-1 network – and a point $y = (F_{i,j}, \lambda_{ij}, \hat{\lambda}_{ij})$ in the space of P1, can verify in polynomial time in N if y is feasible in P1 and if not, it returns a hyperplane that separates y from the feasible region (i.e., an inequality satisfied for the feasible region but not for y).

Our oracle can be divided into two parts: (i) a simple oracle that checks the constraints in (1a)-(1e), and (ii) a more involved oracle for checking the constraint in (1f). Note that since the number of variables and constraints in (1a)-(1e) is polynomial in N , then we can directly check these constraints for y in polynomial time in N . If one constraint is violated, then we return that constraint as the hyperplane that separates y from the feasible set. In what follows, we prove that the constraint in (1f) can also be checked in polynomial time in N . Towards this end, in Section IV-A we first overview some results from [8], and define the M-polytope and the Perfect Matching polytope (PM-polytope) of an undirected graph, and show a useful relationship between them. Note that (similar to the definition of the M-polytope), the PM-polytope of an undirected graph G is the polytope that has all perfect-matchings³ as its extreme points. Then, in Section IV-B we show how a set S that violates the constraint in (1f) can be found by first constructing a Gomory-Hu tree [9] of the weighted graph representing our network, and then checking cuts with a particular structure in it. Finally, in Section IV-C, we show how these results can be leveraged to build our polynomial time separation oracle.

A. M-polytope and PM-polytope

We here overview some results from [8], and show a useful relationship between the M-polytope and the PM-polytope of

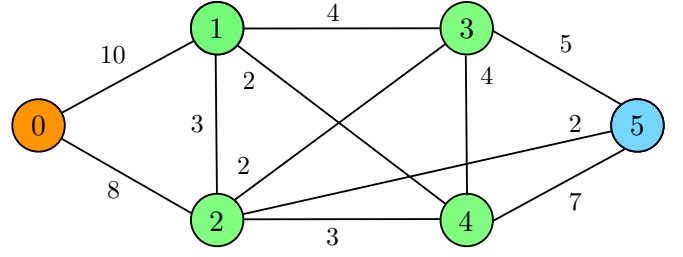


Fig. 3: Example of a weighted graph G .

an undirected graph, which we next define. In particular, we will use the following graph theory notation.

For an undirected graph $G = (V_G, E_G, x_G)$, with set of vertices V_G , set of edges E_G and edge weight function $x_G : E_G \rightarrow \mathbb{R}$, we use the convention $e = (i, j)$ with $i < j$. Furthermore, in the remainder of this section, we use the definitions in Table I.

As an example of the used notation, with reference to the weighted graph G in Fig. 3, we have

$$\begin{aligned} \delta_G(\{1, 3, 4, 5\}) &= \{(0, 1), (1, 2), (2, 3), (2, 4), (2, 5)\}, \\ E_G(\{1, 3, 4, 5\}, \{2\}) &= \{(1, 2), (2, 3), (2, 4), (2, 5)\}, \\ E_G(\{1, 3, 4, 5\}) &= \{(1, 3), (1, 4), (3, 4), (3, 5), (4, 5)\}, \\ x_G(\{(0, 2), (1, 3), (2, 5)\}) &= 8 + 4 + 2 = 14. \end{aligned}$$

We let $G = (V_G, E_G, \hat{\lambda})$ define the weighted undirected graph that describes our Gaussian HD 1-2-1 network, where $\hat{\lambda}(e) = \hat{\lambda}_e$ is defined as in the constraint in (1c) in P1. Then, we note that the constraints in (1d) – (1f) in P1 are those introduced by Edmonds [8] to define the M-polytope for the graph G . By using the notation introduced above, we can rewrite the constraints in (1d) – (1f) in P1, as follows

$$\begin{aligned} \text{M-polytope :} \\ \hat{\lambda}(e) &\geq 0, & \forall e \in E_G, \\ \hat{\lambda}(\delta_G(v)) &\leq 1, & \forall v \in V_G, \\ \hat{\lambda}(E_G(S)) &\leq \frac{|S| - 1}{2}, & \forall S \subseteq V_G, |S| \text{ odd}. \end{aligned} \quad (4)$$

The PM-polytope is represented similarly to the M-polytope, where now the second constraint in (4) is forced to be satisfied with equality, namely

$$\begin{aligned} \text{PM-polytope :} \\ \hat{\lambda}(e) &\geq 0, & \forall e \in E_G, \\ \hat{\lambda}(\delta_G(v)) &= 1, & \forall v \in V_G, \\ \hat{\lambda}(E_G(S)) &\leq \frac{|S| - 1}{2}, & \forall S \subseteq V_G, |S| \text{ odd}. \end{aligned} \quad (5)$$

By using the second constraint in (5), we can manipulate the third constraint, and rewrite the PM-polytope – see Appendix B for the detailed computation – as

$$\begin{aligned} \text{PM-polytope :} \\ \hat{\lambda}(e) &\geq 0, & \forall e \in E_G, \\ \hat{\lambda}(\delta_G(v)) &= 1, & \forall v \in V_G, \\ \hat{\lambda}(E_G(S)) &\geq 1, & \forall S \subseteq V_G, |S| \text{ odd}. \end{aligned} \quad (6)$$

We now show that, for any graph G with N vertices, there is an injection from the M-polytope of G to the PM-polytope

³A perfect matching is a matching such that all vertices in the graph are connected to one edge in the matching set.

TABLE I: Quantities of interest used throughout Section IV.

Quantity	Definition
$x_G(e)$	Weight corresponding to edge $e = (i, j)$ between nodes $i \in V_G$ and $j \in V_G$
$\delta_G(S)$	Set of edges in G that have only one endpoint in the set of vertices $S \subseteq V_G$
$E_G(A, B)$	Set of edges with one endpoint in the set of vertices A and the other endpoint in B with $A, B \subseteq V_G$
$E_G(A)$	Set of edges with both endpoints in the set of vertices A
$x_G(F)$	Sum of the weights of the edges that belong to F , i.e., $x_G(F) = \sum_{e \in F} x_G(e)$

of a constructed graph \tilde{G} with double the number of vertices. Towards this end, we first create a copy $G' = (V_{G'}, E_{G'}, \hat{\lambda}')$ of the original graph G and we let $\tilde{G} = (V_{\tilde{G}}, E_{\tilde{G}}, \tilde{\lambda})$ be a graph with: (i) $V_{\tilde{G}} = V_G \cup V_{G'}$, (ii) $E_{\tilde{G}} = E_G \cup E_{G'} \cup \{(v, v') | v \in V_G, v' \text{ is the copy of } v \text{ in } G'\}$, and (iii) $\tilde{\lambda}(e)$ defined as

$$\tilde{\lambda}(e) = \begin{cases} \hat{\lambda}(e) & \text{if } e \in E_G, \\ \hat{\lambda}'(e) & \text{if } e \in E_{G'}, \\ 1 - \hat{\lambda}(\delta_G(v)) & \text{if } e = (v, v'). \end{cases} \quad (7)$$

We now show that, if $\hat{\lambda}$ is in the M-polytope of G , then $\tilde{\lambda}$ is in the PM-polytope of \tilde{G} . We start by noting that the given construction for $\tilde{\lambda}$ in (7) satisfies the first two constraints in (6). The key challenge is to show that the third constraint is also satisfied, i.e.,

$$\tilde{\lambda}(\delta_{\tilde{G}}(S)) \geq 1, \quad \forall S \subseteq V_{\tilde{G}}, |S| \text{ odd}. \quad (8)$$

In Appendix C, we show that the constraint in (8) is indeed also satisfied. This result implies that we can check whether $\hat{\lambda}(e)$ is in the M-polytope of G by checking whether $\tilde{\lambda}(e)$, with the construction in (7), is in the PM-polytope of \tilde{G} .

B. PM-polytope and Gomory-Hu tree

In the previous subsection, we have defined the PM-polytope for the weighted undirected graph \tilde{G} as in (6). In particular, in (6) we can check in polynomial time if the first two sets of constraints are satisfied. Therefore, our main concern lies in the third group of constraints, since there is an exponential number of them. We here show that a set S that violates the third constraint in (6) can be found by first constructing a Gomory-Hu tree [9] of \tilde{G} (defined below), and then checking cuts with a particular structure in it.

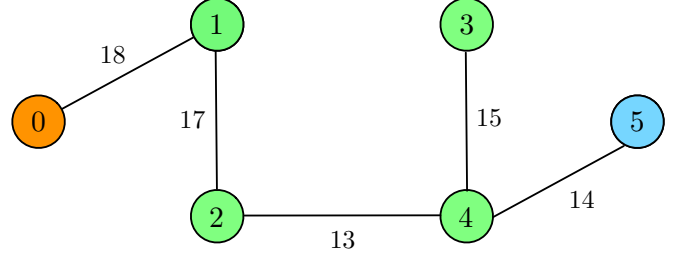
We start by noting that the third group of constraints in (6) can be written in a compact way as

$$\min_{\substack{S \subseteq V_{\tilde{G}}, \\ |S| \text{ odd}}} \tilde{\lambda}(\delta_{\tilde{G}}(S)) \geq 1. \quad (9)$$

In words, what this says is that the *minimum odd* cut in \tilde{G} has a value greater than or equal to 1. An *odd* cut is a vertex partition of $V_{\tilde{G}}$ into S and S^c such that either S and/or S^c has an odd cardinality (S^c is the complement of S).

In [11], Padberg and Rao provided an efficient algorithm to find the minimum odd cut and its value for any graph G . An appealing feature of the algorithm designed in [11] is that it runs in polynomial time in $|V_G|$. In particular, the method introduced consists of using *Gomory-Hu* trees.

Definition 1 (Gomory-Hu Tree). Let $G = (V_G, E_G, x_G)$ be a capacitated (weighted) undirected graph with capacity


 Fig. 4: A Gomory-Hu tree T of the graph G in Fig. 3.

function $x_G : E_G \rightarrow \mathbb{R}_+$. A *Gomory-Hu* tree (for G and x_G) is a capacitated tree $T = (V_G, F, \alpha_T)$ with capacity α_T , such that for each edge $e = (s, t) \in F$, the two components of $T \setminus e$ give a minimum capacity $s - t$ cut in G . The capacity of the cut in G is equal to $\alpha_T(s, t)$.

Note that, for any capacitated undirected graph with N vertices, a Gomory-Hu tree always exists and can be constructed by the algorithm in [9] using $|V_G| - 1$ runs of the max-flow problem. Given the triangle inequality of min-cuts in a graph, Definition 1 implies the following property of Gomory-Hu trees.

Property 1. Let $G = (V_G, E_G, x_G)$ be a capacitated undirected graph and $T = (V_G, F, \alpha_T)$ be a Gomory-Hu tree of G . Consider any two vertices $u, v \in V_G$, let \mathcal{P}_{uv} be the path connecting u and v in T and let (s, t) be the edge with the minimum capacity $\alpha_T(s, t)$ along the path \mathcal{P}_{uv} . Then, we have the two following properties:

- 1) The two components $T \setminus (s, t)$ give a minimum capacity $u - v$ cut in G ;
- 2) The value of the minimum capacity $u - v$ cut is given by $\alpha_T(s, t)$.

As an example of a Gomory-Hu tree, consider Fig. 4, which represents a Gomory-Hu tree T of the graph G in Fig. 3. From Property 1, it follows that if for G in Fig. 3, we would like to know the minimum cut between vertices 1 and 5, then all we need to do is to look at the unique path connecting 1 and 5 in T in Fig. 4. The edge with the minimum capacity is $(2, 4)$ and hence the min-cut between 1 and 5 equals 13. A cut that has this value in G is the partition $U = \{0, 1, 2\}$ and $U^c = \{3, 4, 5\}$.

We now state a simplified version of the result proved in [11] that shows how a Gomory-Hu tree of a graph can be leveraged to verify if the minimum odd cut of a graph satisfies (9). This theorem is proved in Appendix D.

Theorem 4. Let $\tilde{G} = (V_{\tilde{G}}, E_{\tilde{G}}, \tilde{\lambda})$ be an undirected capacitated graph with $|V_{\tilde{G}}|$ even, and let $T = (V_{\tilde{G}}, F, \alpha_T)$ be a

Gomory-Hu tree for \tilde{G} . Then, one of the cuts determined by $T \setminus e$, $\forall e \in F$, is a minimum capacity odd cut in \tilde{G} .

Thus, we can use the result in Theorem 4 to verify whether the minimum odd cut has a weight greater than or equal to one, by following the procedure illustrated in Algorithm 1. We note that in Algorithm 1, even though we need only one among $|W_f|$ and $|W_f^c|$ to be odd, in a graph with even number of vertices, if one is odd, then also the other is odd. This explains why we check if both $|W_f|$ and $|W_f^c|$ are odd.

In summary, we have here shown that we can find a set S (if any) that violates the third constraint of the PM-polytope of $\tilde{G} = (V_{\tilde{G}}, E_{\tilde{G}}, \hat{\lambda})$ defined in (6) by first constructing a Gomory-Hu tree $T = (V_{\tilde{G}}, F, \alpha_T)$ and then checking odd cuts in it. Since the number of cuts in T is $|V_{\tilde{G}}| - 1 = 2N + 3$, then by using this procedure we only need to perform $O(N)$ checks (and not an exponential number of them).

Algorithm 1 Check if minimum odd cut in \tilde{G} satisfies (9)

```

1: function CHECKMINIMUMODDCUT( $\tilde{G}$ )
2:   Build a Gomory-Hu tree  $T = (V_{\tilde{G}}, F, \alpha_T)$  of  $\tilde{G}$ 
3:   for each  $f \in F$  do
4:     Let  $W_f$  and  $W_f^c$  be the two components of  $T \setminus f$ 
5:     if  $|W_f|$  and  $|W_f^c|$  are odd then
6:       if  $\alpha_T(f) < 1$  then
7:         return  $W_f$  as the set that violates (9)
   return  $W_f = \emptyset$ 

```

C. Polynomial time separation oracle

We here show how the results that we discussed and stated in the previous two subsections can be combined and leveraged to build our polynomial time separation oracle. In particular, Algorithm 2 provides the pseudocode of our separation oracle. It is worth noting that each step in Algorithm 2 can be performed in polynomial time in N and hence our oracle runs in polynomial time in N . In particular, the complexity is dominated by the step where a Gomory-Hu tree is constructed for \tilde{G} . This construction algorithm, in fact, involves performing the max-flow problem $2N + 3$ times and hence the complexity of this step – and consequentially of our oracle – is $O(N^4)$ in the worst-case. This concludes the proof of Theorem 3.

V. FINDING A SCHEDULE IN POLYNOMIAL-TIME

To prove part (b) of Theorem 1, we first note that, for a Gaussian HD 1-2-1 network, a state s in (3b) does not activate two adjacent links. Thus, a state is a matching of directed edges in a directed graph representing the network topology. Now, assume that we are given a feasible point in the LP P1 (obtained by solving P1 as described in Section IV). The main objective of this section is to efficiently construct a set of matchings (representing states in the network) and find their corresponding activation times (representing λ_s in (3b)), such that the fraction of time a link $i \rightarrow j$ is active is equal to λ_{ji} in P1. For any pair of nodes $i < j$, we refer to $\hat{\lambda}_{ji}$ in P1 as the *connection* activation time, i.e., $\hat{\lambda}_{ji}$ represents the duration of time nodes i and j are connected,

Algorithm 2 Polynomial time separation oracle for P1

Input: Network NET, point y to test for feasibility

Output: Feasible flag, violated constraint in P1

```

1: Check if all constraints in (1a)-(1e) in P1 are satisfied
2: if one constraint in (1a)-(1e) is violated then
3:   return Feasible = False, constraint violated
4: Construct undirected simple graph  $G = (V_G, E_G, \hat{\lambda})$ 
   with the same set of nodes in NET, edges representing
   links in NET (but only one direction) and  $\hat{\lambda}$  given by  $y$ 
5: Construct undirected graph  $\tilde{G}$  from  $G$  that has double
   number of vertices as described in Section IV-A
6:  $W_f = \text{CHECKMINIMUMODDCUT}(\tilde{G})$ 
7: if  $W_f = \emptyset$  then
8:   return Feasible = True
9: else
10:  Feasible = False
11:  Let  $W_{f,a} = W_f \cap V_G$  and  $W'_{f,b} = W_f \cap V_{G'}$ 
12:  Let  $W'_{f,a}$  be a copy of  $W_{f,a}$  in  $V_{G'}$ 
13:  Let  $W_{f,b}$  be a copy of  $W'_{f,b}$  in  $V_G$ 
14:  if  $|W_{f,a} \setminus W_{f,b}|$  odd then
15:     $Z = W_{f,a} \setminus W_{f,b}$ 
16:  else
17:     $Z = W'_{f,b} \setminus W'_{f,a}$ 
18:  return Feasible = False, constraint  $Z$  in (1f) violated

```

without considering the direction of communication between them. Thus, from a connection activation time perspective, the network is represented by an undirected graph where an edge (i, j) is active for a fraction $\hat{\lambda}_{ji}$ of time. We first discuss how we can decompose the connection activation times into undirected matchings, and then show how these can be leveraged to construct our set of directed matchings (states). The goal is to show that both these tasks can be performed in polynomial-time in the number of relays N .

A. Decomposition into undirected matchings

We define the undirected graph $G = (V_G, E_G, \hat{\lambda})$, where: (i) the graph vertices in V_G represent the nodes in our Gaussian HD 1-2-1 network, (ii) $E_G = \{(i, j) | i > j, \hat{\lambda}_{ij} > 0\}$ is the set of edges, and (iii) the edge weights are equal to the values of $\hat{\lambda}_{ij}$ from the feasible point in P1. Note that, without loss of generality, in the definition of E_G we do not include any edge e for which $\hat{\lambda}_e = 0$.

Let $\hat{\lambda} \in \mathbb{R}^{|E_G|}$ be the vector comprised of $\hat{\lambda}_e$, $\forall e \in E_G$. As highlighted in Section IV, the constraints on $\{\hat{\lambda}_e\}$ in (1e) – (1g) describe the M-polytope of the undirected graph G [8]. Our goal here is to efficiently find a set of K matchings $M_k \in \mathbb{R}^{|E_G|}$ (vertices of the M-polytope) such that

$$\hat{\lambda} = \sum_{k=1}^K \varphi_k M_k, \quad \sum_{k=1}^K \varphi_k = 1, \quad \varphi_k \geq 0, \quad \forall k \in [1:K]. \quad (10)$$

By Caratheodory's theorem [12], we know that for some $K \leq |E_G| + 1$, such a decomposition of $\hat{\lambda}$ exists. However, the key challenge is to discover this decomposition in polynomial-time in N . Towards this end, we appeal to a

result in combinatorial optimization [13, Theorem 6.5.11]. This theorem states that, if we can optimize an objective function over the M-polytope using a separation oracle that runs in polynomial-time, then an algorithmic implementation of Caratheodory's theorem can be performed in polynomial-time in the number of variables. Our result in Theorem 3 proves that such a polynomial-time separation oracle exists, and hence [13, Theorem 6.5.11] ensures that the decomposition in (10) can be performed in polynomial-time.

The remainder of this subsection is devoted to describing the decomposition algorithm [13, Theorem 6.5.11] and explaining why we can apply it to our M-polytope. For this, we need to explicitly mention some properties of polyhedra.

Definition 2. The dimension of a polyhedron $P \subseteq \mathbb{R}^n$, $\dim(P)$ is the maximum number of affinely independent points in P minus 1. If $\dim(P) = n$, we say that P is **fully-dimensional**. A polyhedron P is said to be **bounded** if there exists a ball B in \mathbb{R}^n centered around the origin with radius $r < \infty$ such that $P \subseteq B$.

Definition 3. A polyhedron $P \subseteq \mathbb{R}^n$ is called **rational** if all its vertices and at least one point in its interior belong to \mathbb{Q}^n . A polyhedron P is called **well-described**, if a finite number of bits is needed to encode a single constraint of the polyhedron.

Definition 4. The subset F is called a **face** of polyhedron $P \subseteq \mathbb{R}^n$, if there exists an inequality such that $a^T x \leq a_0$, $\forall x \in P$ and $F = \{x \in P \mid a^T x = a_0\}$. We say that the inequality $a^T x \leq a_0$ defines the face F .

Definition 5. A face F of polyhedron $P \subseteq \mathbb{R}^n$, is called a **facet** of P if $\dim(F) = \dim(P) - 1$. If the polyhedron P is fully-dimensional, then each non-redundant inequality constraint of P defines a facet of P .

It is not difficult to see that the M-polytope is rational, well-described and fully-dimensional. To see full-dimensionality, note that the set of matchings (vertices) such that each selects only one edge in the graph, together with the all-zero matching, form a set of affinely independent points with $|E_G| + 1$ elements. As a result (by Definition 5) each constraint in (1e) – (1g) defines a facet of the M-polytope.

To describe the decomposition algorithm for our polytope, we need to shorthand two abstract oracles that we will use with their shorthand (SEP and OPT), extensively.

- 1) $SEP(P, y)$ denotes the separation oracle that takes a polyhedron P and a point y . This oracle determines if $y \in P$ or else returns a constraint of P that is violated by y . If P is fully-dimensional, then this returned constraint defines a facet of P as aforementioned above. Note that P might not be explicitly defined to the oracle with a set of constraints, but rather as an object (e.g., a graph) and a condition on the object (e.g., minimum odd cut is greater than some value). An example of such an oracle is the result in Theorem 3.
- 2) $OPT(P, c)$ denotes the optimization oracle that given a polyhedron P and an affine objective function parameterized with c , maximizes $c^T x$ over $x \in P$. If P is

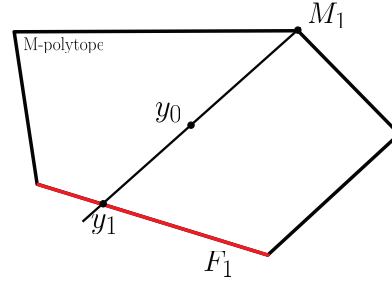


Fig. 5: An example of the decomposition of y_0 into M_1 and y_1 . The red color represents the facet F_1 that y_1 belongs to.

bounded then for any c , the oracle returns a vertex of P at which $c^T x$ is maximized.

We are now ready to describe the algorithm used for the decomposition in (10) over the M-polytope. The exact details are described in Algorithm 3. The skeleton of Algorithm 3 is the following:

- 1) We start with our desired point to decompose $y_0 = \hat{\lambda}$ and find any vertex M_1 of the M-polytope (using OPT).
- 2) Maximize along the line connecting y_0 and M_1 in the M-polytope in the direction $y_0 - M_1$. The maximizer, denoted as y_1 , is the intersection of the line and some boundary of the M-polytope. A simple illustration of y_0 , y_1 and M_1 is shown in Fig. 5.
- 3) Since y_0 is in between M_1 and y_1 on a line, then we can write $y_0 = \theta_1 M_1 + (1 - \theta_1) y_1$ for $0 \leq \theta \leq 1$.
- 4) Next, we find a facet F_1 (defined as $a_1^T x = b_1$) of the M-polytope containing y_1 by finding the constraint separating $y_1 + \varepsilon(y_0 - M_1)$ from the M-polytope for some $\varepsilon > 0$ (using SEP).
- 5) We want to find a vertex of this facet so that we maximize the objective function $c_1 = a_1$ (using OPT) to get a vertex M_2 in the F_1 .
- 6) Since both y_1 and M_2 belong to F_1 then the line projected along them to get y_2 also belongs to F_1 .
- 7) Repeat Steps 2) to 6) for $i > 1$ until we hit a vertex in the end. Note that in Step 5) to get M_{i+1} that belongs to the intersection of all facets F_1, F_2, \dots, F_i , we use $c_i = \sum_{j=1}^i a_i$ to ensure that the vertex satisfies $a_j^T x = b_j$, $\forall j \in [1 : i]$.
- 8) At the end, we have that for each $i \in [1 : |E_G| + 1]$,

$$y_{i-1} = \theta_i y_i + (1 - \theta_i) M_i, \quad 0 \leq \theta_i \leq 1.$$

Thus, by applying recursion, we can express our desired point y_0 as

$$y_0 = \sum_{i=1}^{|E_G|+1} \varphi_i M_i,$$

with

$$\varphi_i = \theta_i \prod_{j=1}^{i-1} (1 - \theta_j).$$

Note that Algorithm 3 above iterates over the oracles for a polynomial number of times equal to $|E_G| + 1$, since at

Algorithm 3 Decomposition into Caratheodory points

Input: Graph $G = (V_G, E_G, \hat{\lambda})$

Output: $\{\varphi_i\}, \{M_i\}$

```

1:  $y_0 = \hat{\lambda}$ 
2:  $P = \text{M-polytope of } G$ 
3:  $c_0 = 1_{|E_G| \times 1}$  // Set first objective function to any value (all ones here)
4: for each  $i \in [1 : |E_G| + 1]$  do
5:    $M_i = \text{OPT}(P, c_{i-1})$  // Get vertex maximizing  $c_{i-1}$ 
6:   // Maximize in the direction of  $y_{i-1} - M_i$  to get the point  $y_i$  on the boundary of  $P$ 
7:    $y_i = \text{OPT}(P \cap \text{line}(y_{i-1}, M_i), y_{i-1} - M_i)$ 
8:   // Convex combination of  $M_i$  and  $y_i$  gets  $y_{i-1}$ 
9:    $\theta_i = \frac{M_i - y_i}{y_{i-1} - y_i}$  // Solve for  $0 \leq \theta_i \leq 1 : \theta_i M_i + (1 - \theta_i) y_i = y_{i-1}$ 
10:  // Get the facet containing  $y_i$ . Perturb  $y_i$  by a small amount in the direction of  $y_{i-1} - M_i$  to get a point outside of
  the polyhedron, and then apply the separation oracle.
11:   $a_i^T, b_i = \text{SEP}(P, y_i + \varepsilon \frac{y_{i-1} - M_i}{|y_{i-1} - M_i|})$  //  $a_i^T x = b_i$  defines a facet containing  $y_i$ 
12:  // We want the new vertex in the next iteration to be in the intersection of all facets  $F_j$  visited before. This is done
  by adding all the inequalities defining these facets
13:   $c_i = \sum_{j=1}^i a_j$ 
14: for each  $i \in [1 : |E_G| + 1]$  do
15:   $\varphi_i = \theta_i \prod_{j=1}^{i-1} (1 - \theta_j)$ 

```

each time, we are restricting our search with a new equation ($a_j^T x = b_j$), i.e., at every iteration we decrease the dimension of the currently considered polytope by 1. Therefore, if the oracles SEP and OPT can run in polynomial time, then the algorithm is polynomial in time. Fortunately, the M-polytope has a polynomial-time separation oracle (by Theorem 3) and a polynomial-time optimization oracle (by Theorem 1). Thus, by consequence Algorithm 2 runs in polynomial-time in the number of relay nodes N .

B. Post-processing for directional matchings

We now need to utilize the matchings $\{M_k\}$ and their activation times $\{\varphi_k\}$ output by the algorithm discussed in the previous subsection to construct network states and find their activation times such that each link $i \rightarrow j$ is activated for a duration λ_{ji} (output by P1). We can perform this decomposition in polynomial-time by iterating over the edges of the undirected graph G constructed in the previous subsection. For notational ease, we can rewrite each of the matchings M_k discovered in the previous subsection as a lower triangular matrix $\widetilde{M}_k \in \mathbb{R}^{(N+2) \times (N+2)}$, where

$$\widetilde{M}_k(j, i) = \begin{cases} M_k(e) & (i-1, j-1) \in E_G, i < j \\ 0 & \text{otherwise,} \end{cases}$$

where $i-1$ and $j-1$ are used since the matrix entries are matched with positive numbers while our nodes are indexed from 0. The rows of \widetilde{M}_k represent the receiving modules of the nodes, while the column indexes represent the transmitting modules of the nodes. As an illustrative example, consider a Gaussian HD 1-2-1 network with $N = 2$ relays, where the undirected graph G has edge set

$E_G = \{(0, 1), (0, 2), (1, 2), (1, 3), (2, 3)\}$. For the matching M_k shown below, we have the corresponding \widetilde{M}_k

$$M_k = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \implies \widetilde{M}_k = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

With this notation, we can design an algorithm that generates the $\{\lambda_s\}$ optimal for (3b) such that they collectively activate each link $i \rightarrow j$ for the duration λ_{ji} output by the LP P1. The formal procedure of the algorithm is described in Algorithm 4. The main idea behind Algorithm 4 is the following. From (10), the definition of \widetilde{M}_k and the LP P1, we know that $\forall (i, j) \in [0 : N+2]$, such that $i < j$, we have

$$\sum_{k=1}^{|E_G|+1} \varphi_k \widetilde{M}_k(j, i) = \hat{\lambda}_{ij} \stackrel{\text{(P1)}}{=} \lambda_{ji} + \lambda_{ij}.$$

Thus, for each connection (i, j) in the network, we just need to break the matchings $\{\widetilde{M}_k\}$ into two sets contributing to the activation of the links $i \rightarrow j$ and $j \rightarrow i$. Before processing a connection (i, j) , the default direction is $i \rightarrow j$. The algorithm iterates over each connection (i, j) , and adds up the activation times for the matchings one by one until the sum exceeds λ_{ji} . For the remaining matchings, we change the assigned direction to $j \rightarrow i$. The matching $\widetilde{M}_k(j, i)$ that caused the sum to exceed λ_{ji} is split into two copies, one where the direction is $i \rightarrow j$ and the other with $j \rightarrow i$.

Note that, in each iteration over the elements in E_G , we split at most one matching (state). Thus, starting with $|E_G|+1$ matchings, we end up with at most $2|E_G|+1$ matchings. Moreover, the inner loop iterates over at most $2|E_G|+1$ matchings. Thus, the algorithm runs in $O(|E_G|^2)$ time which in the worst case is $O(N^4)$ for a network with N relays.

Algorithm 4 Constructing digraph matchings from undirected graph matchings

Input: Graph $G = (V_G, E_G, \hat{\lambda})$, $\{\widetilde{M}_k\}_{k=1}^{|E_G|+1}$, $\{\varphi\}_{k=1}^{|E_G|+1}$

Output: $\{\varphi'_k\}$, $\{\widetilde{M}'_k\}$

```

1:  $K = |E_G| + 1$ 
2:  $\widetilde{M}'_k = \widetilde{M}_k$ ,  $\varphi'_k = \varphi_k$ ,  $\forall k \in [1 : |E_G| + 1]$ 
3: for each  $(i, j) \in E_G$  do
4:    $\omega = 0$ 
5:   for each  $k \in [1 : K]$  do
6:     if  $\widetilde{M}'_k(j, i) == 1$  then
7:       if  $\omega + \varphi'_k < \lambda_{ji}$  then // All matchings that contain  $(i, j)$  connection so far are less than  $\lambda_{ji}$ 
8:          $\omega = \omega + \varphi'_k$ 
9:       else if  $\omega + \varphi'_k = \lambda_{ji}$  then // All matchings that contain  $(i, j)$  connection so far are exactly  $\lambda_{ji}$ 
10:        // For all remaining matchings containing  $(i, j)$  assign to  $j \rightarrow i$ 
11:        for each  $\ell \in [k + 1 : K]$  s.t.  $\widetilde{M}'_\ell(j, i) == 1$  do
12:           $\widetilde{M}'_\ell(j, i) = 0$ ,  $\widetilde{M}'_\ell(i, j) = 1$ 
13:        Break
14:      else
15:        // Make another copy ( $K + 1$ ) of the current state, and assign  $j \rightarrow i$  instead of  $i \rightarrow j$ 
16:         $\widetilde{M}'_{K+1} = \widetilde{M}'_k$  // Make a new copy at the end of the list
17:         $\varphi'_{K+1} = \omega + \varphi'_k - \lambda_{ji}$ ,  $\varphi'_k = \lambda_{ji} - \omega$ 
18:         $\widetilde{M}'_{K+1}(j, i) = 0$ ,  $\widetilde{M}'_{K+1}(i, j) = 1$ 
19:        // For all remaining matchings in  $[k + 1 : K]$  containing  $(i, j)$  assign to  $j \rightarrow i$ 
20:        for each  $\ell \in [k + 1 : K]$  s.t.  $\widetilde{M}'_\ell(j, i) == 1$  do
21:           $\widetilde{M}'_\ell(j, i) = 0$ ,  $\widetilde{M}'_\ell(i, j) = 1$ 
22:         $K = K + 1$  // Increase the number of states by one due to the copy created in line 16
23:      Break
```

APPENDIX A
PROOF OF THEOREM 2

We start this section by taking note of an observation in [2], which states that, for a fixed λ_s , the inner minimization in $C_{cs, \text{iid}}$ in (3b) is the standard min-cut problem over a graph with link capacities given by

$$\ell_{j,i}^{(s)} = \left(\sum_{s: j \in s_{i,t}, i \in s_{j,r}} \lambda_s \right) \ell_{j,i}. \quad (11)$$

We can therefore leverage the fact that the dual of the min-cut problem is the max-flow problem, and use strong duality to replace the inner minimization in (3b) with the max-flow problem over the graph with link capacities defined in (11). With this, we obtain

$$\begin{aligned}
\text{P0 : } C_{cs, \text{iid}} &= \max \sum_{j=1}^{N+1} F_{j,0} \\
(0a) \quad 0 &\leq F_{j,i} \leq \ell_{j,i}^{(s)}, \quad \forall (i, j) \in [0:N] \times [1:N+1], \\
(0b) \quad \sum_{j \in [1:N+1] \setminus \{i\}} F_{j,i} &= \sum_{k \in [0:N] \setminus \{i\}} F_{i,k}, \quad \forall i \in [1:N], \\
(0c) \quad \sum_s \lambda_s &\leq 1, \\
(0d) \quad \lambda_s &\geq 0,
\end{aligned}$$

where $F_{j,i}$ represents the flow from node i to node j , and where the constraint in (0b) ensures that flow is conserved

at each relay node. Thus, to prove Theorem 2, we need to show that the linear program P1 is equivalent to the linear program P0.

We start by noting that all variables, except λ_s and λ_{ij} , are the same in the two linear programs P0 and P1. Thus, to make P0 and P1 equivalent, we want to find a bijective mapping between λ_s 's and λ_{ij} 's.

If we observe a state s in a Gaussian HD 1-2-1 network and hence in P0, we see that a state does not activate two adjacent links, i.e., a state represents a matching⁴ in the graph representing the network topology. Furthermore, the constraints in (0a), (0c) and (0d) in P0 suggest that the link activation times should be in the convex hull of the 0-1 activations representing a matching. To state this more formally, let F be a matrix that is populated by the flow variables $F_{j,i}$ in P0, and let L be a matrix populated by the link capacities $\ell_{j,i}$ in P0. For each state s , let M_s be a binary matrix such that its (j, i) -th component is 1 if the link from node i to node j is activated by state s , and 0 otherwise. Then, we can write the constraints in (0a) and (0c) in P0 as

$$(0a) : F \leq \left(\sum_s \lambda_s M_s \right) \odot L, \quad (12a)$$

$$(0c) : \sum_s \lambda_s = 1, \quad (12b)$$

⁴A matching in a graph is a set of graph edges such that each vertex (node) in the graph has at most one edge from the set connected to it.

where the operator \odot denotes the element-wise multiplication (Hadamard product). Note that we made the sum of the state probabilities in (12b) equal to 1, and this is without loss of generality. This is because one valid M_s is populated by all zeros, and hence we can associate to this all-zero matrix a weight so that the sum equals 1, without changing the main inequality in (12a). Now, if we consider P1, then the constraint in (1a) can be written as

$$F \leq \Lambda \odot L, \quad (13)$$

where Λ is populated by the variables λ_{ji} .

From (12) and (13), we can see that the constraint needed in P1 is to have Λ in the convex hull of $\{M_s\}$, i.e., we want the link activations (or weights) that are in the convex hull of points representing matchings in a graphs. A result [8] by Edmonds for undirected graphs characterizes the constraints that represent the Matching polytope (M-polytope). The M-polytope of an undirected graph G is the polytope that has all matchings as its extreme points. By massaging this result to apply to the directed graph in our problem, we define $\hat{\lambda}_{ij} = \lambda_{ij} + \lambda_{ji}$ for all $i < j$ and we apply Edmonds's constraints on $\hat{\lambda}_{ij}$. With this, we precisely get the linear program in P1. In particular, the M-polytope characterized by Edmonds defines the constraints in (1e) and (1f) on $\hat{\lambda}_{ij}$. The mapping from $\hat{\lambda}$ to λ_{ij} is defined by the constraint in (1c) in P1. This concludes the proof of Theorem 2.

APPENDIX B

EQUIVALENCE BETWEEN PM-POLYTOPES IN (5) AND (6)

We note that in (5) and (6), the first two constraints are the same. Thus, what we need to show is the equivalence between the third constraints in (5) and (6). In other words, given an undirected graph $G = (V_G, E_G, \hat{\lambda})$, with

$$\begin{aligned} \hat{\lambda}(e) &\geq 0, \quad \forall e \in E_G, \\ \hat{\lambda}(\delta_G(v)) &= 1, \quad \forall v \in V_G, \end{aligned} \quad (14)$$

we want to show that for $\forall S \subseteq V_G, |S|$ odd, we have that

$$\hat{\lambda}(E_G(S)) \leq \frac{|S| - 1}{2} \iff \hat{\lambda}(\delta_G(S)) \geq 1.$$

A key property used to prove both directions is the fact that for any $S \subseteq V_G$, we have – by simple counting – that

$$\sum_{v \in S} \hat{\lambda}(\delta_G(v)) = \hat{\lambda}(\delta_G(S)) + 2\hat{\lambda}(E_G(S)). \quad (15)$$

\implies direction: Let S be an arbitrary subset of V_G with odd cardinality. Then, we have that

$$\begin{aligned} |S| &\stackrel{(14)}{=} \sum_{v \in S} \hat{\lambda}(\delta_G(v)) \\ &\stackrel{(15)}{=} \hat{\lambda}(\delta_G(S)) + 2\hat{\lambda}(E_G(S)) \\ &\geq \hat{\lambda}(\delta_G(S)) + 2\frac{|S| - 1}{2}. \end{aligned}$$

The above expression can be rewritten to give that $\hat{\lambda}(\delta_G(S)) \geq 1$, which is precisely what we intended to show.

\Leftarrow direction: Let S be an arbitrary subset of V_G with odd cardinality. Then, we have that

$$\begin{aligned} |S| &\stackrel{(14)}{=} \sum_{v \in S} \hat{\lambda}(\delta_G(v)) \\ &\stackrel{(15)}{=} \hat{\lambda}(\delta_G(S)) + 2\hat{\lambda}(E_G(S)) \\ &\geq 1 + 2\hat{\lambda}(E_G(S)), \end{aligned}$$

which, when reorganized, gives that $\hat{\lambda}(E_G(S)) \leq (|S| - 1)/2$. This concludes the proof that (5) and (6) are equivalent.

APPENDIX C

PROOF THAT $\tilde{\lambda}$ IN (7) SATISFIES (8)

We want to show that given $\hat{\lambda}$ in the M-polytope of G , then $\tilde{\lambda}$ constructed as in (7) satisfies (8) – and hence it is in the PM-polytope of \tilde{G} .

For any $S \subseteq V_{\tilde{G}}$ we define $S_a = S \cap V_G$ and $S'_b = S \cap V_{G'}$. We let S'_a be the copy of S_a in $V_{G'}$ and similarly, we let S_b be the copy of S'_b in V_G . Note that, $S = S_a \cup S'_b$. Fig. 6 provides an illustration of these sets. We now consider two different cases for S with odd cardinality and, for each of them, prove that the constraint in (8) is indeed satisfied. This will show that the construction of $\tilde{\lambda}$ in (7) satisfies the constraint in (8).

- 1) **Case 1:** $S'_b = \emptyset$. In this case, since $S = S_a \cup S'_b$, then $|S| = |S_a|$ is odd, and we have that

$$\begin{aligned} \tilde{\lambda}(\delta_{\tilde{G}}(S)) &= \tilde{\lambda}(\delta_{\tilde{G}}(S_a)) \\ &\stackrel{(a)}{=} \sum_{v \in S_a} \tilde{\lambda}(\delta_{\tilde{G}}(v)) - 2\tilde{\lambda}(E_{\tilde{G}}(S_a)) \\ &\stackrel{(b)}{=} |S_a| - 2\tilde{\lambda}(E_{\tilde{G}}(S_a)) \\ &\stackrel{(c)}{\geq} |S_a| - 2\frac{|S_a| - 1}{2} = 1, \end{aligned}$$

where: (i) the equality in (a) follows by using counting arguments (see also (15) in Appendix B); (ii) the equality in (b) follows from the fact that our construction of $\tilde{\lambda}$ satisfies the second constraint in (6); (iii) the inequality in (c) follows from the fact that, by our construction in (7) $\tilde{\lambda}(E_{\tilde{G}}(S_a)) = \hat{\lambda}(E_G(S_a))$ and for any $S_a \subseteq V_G$, from the third constraint in (4), we have that $\hat{\lambda}(E_G(S_a)) \leq \frac{|S_a| - 1}{2}$. Thus, for this case the constraint in (8) is satisfied.

- 2) **Case 2:** $S'_b \neq \emptyset$. We start by noting that (see also Fig. 6)

$$\begin{aligned} |S| &= |S_a| + |S'_b| \\ &= |S_a \setminus S_b| + \underbrace{|S_a \cap S_b|}_Z + |S'_b \setminus S'_a| + \underbrace{|S'_b \cap S'_a|}_{Z'} \\ &= |S_a \setminus S_b| + |S'_b \setminus S'_a| + 2|S_a \cap S_b|. \end{aligned}$$

Thus, since $|S|$ is odd, then one (and only one) among $|S_a \setminus S_b|$ and $|S'_b \setminus S'_a|$ is odd. Without loss of generality,

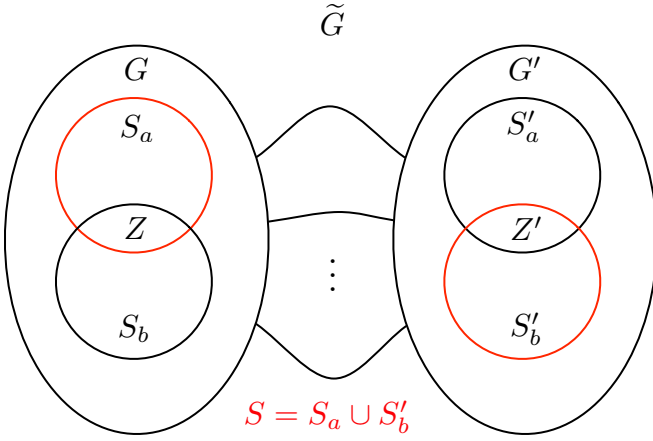


Fig. 6: Illustration for the different sets used in Appendix C.

assume that $|S_a \setminus S_b|$ is odd. Then, we have that (see also Fig. 6)

$$\begin{aligned}
 \tilde{\lambda}(\delta_{\tilde{G}}(S)) &= \tilde{\lambda}(\delta_{\tilde{G}}(S_a \setminus S_b)) + \tilde{\lambda}(\delta_{\tilde{G}}(Z)) \\
 &\quad - 2\tilde{\lambda}(E_{\tilde{G}}(S_a \setminus S_b, Z)) - \tilde{\lambda}(E_{\tilde{G}}(Z, Z')) \\
 &\quad + \tilde{\lambda}(\delta_{\tilde{G}}(S'_b \setminus S'_a)) + \tilde{\lambda}(\delta_{\tilde{G}}(Z')) \\
 &\quad - 2\tilde{\lambda}(E_{\tilde{G}}(S'_b \setminus S'_a, Z')) - \tilde{\lambda}(E_{\tilde{G}}(Z, Z')) \\
 &\stackrel{(a)}{=} \tilde{\lambda}(\delta_{\tilde{G}}(S_a \setminus S_b)) + \tilde{\lambda}(\delta_{\tilde{G}}(S'_b \setminus S'_a)) \\
 &\quad + \tilde{\lambda}(\delta_{\tilde{G}}(Z)) - \beta^* \\
 &\quad + \tilde{\lambda}(\delta_{\tilde{G}}(Z')) - \beta^* \\
 &\stackrel{(b)}{=} \tilde{\lambda}(\delta_{\tilde{G}}(S_a \setminus S_b)) + \tilde{\lambda}(\delta_{\tilde{G}}(S'_b \setminus S'_a)) \\
 &\quad + 2\tilde{\lambda}(\delta_{\tilde{G}}(Z)) - 2\gamma^* \\
 &\geq \tilde{\lambda}(\delta_{\tilde{G}}(S_a \setminus S_b)) + \tilde{\lambda}(\delta_{\tilde{G}}(S'_b \setminus S'_a)),
 \end{aligned}$$

where: (i) the equality in (a) follows by defining

$$\begin{aligned}
 \beta^* &= \left[\tilde{\lambda}(E_{\tilde{G}}(S'_b \setminus S'_a, Z')) + \tilde{\lambda}(E_{\tilde{G}}(Z, Z')) \right. \\
 &\quad \left. + \tilde{\lambda}(E_{\tilde{G}}(S_a \setminus S_b, Z)) \right];
 \end{aligned}$$

(ii) the equality in (b) follows by noting that G' is a copy of G and hence $\tilde{\lambda}(\delta_{\tilde{G}}(Z)) = \tilde{\lambda}(\delta_{\tilde{G}}(Z'))$, $\tilde{\lambda}(E_{\tilde{G}}(S'_a \setminus S'_b, Z')) = \tilde{\lambda}(E_{\tilde{G}}(S_a \setminus S_b, Z))$ and similarly, $\tilde{\lambda}(E_{\tilde{G}}(S'_b \setminus S'_a, Z')) = \tilde{\lambda}(E_{\tilde{G}}(S_b \setminus S_a, Z))$; thus, with this, we have

$$\begin{aligned}
 \beta^* &= \gamma^* = \left[\tilde{\lambda}(E_{\tilde{G}}(S_a \setminus S_b, Z)) + \tilde{\lambda}(E_{\tilde{G}}(Z, Z')) \right. \\
 &\quad \left. + \tilde{\lambda}(E_{\tilde{G}}(S_b \setminus S_a, Z)) \right].
 \end{aligned}$$

From the series of inequalities above, we have that

$$\tilde{\lambda}(\delta_{\tilde{G}}(S)) \geq \tilde{\lambda}(\delta_{\tilde{G}}(S_a \setminus S_b)) + \tilde{\lambda}(\delta_{\tilde{G}}(S'_b \setminus S'_a)). \quad (16)$$

Since we assumed, without loss of generality, that $|S_a \setminus S_b|$ is odd and by using similar arguments as in Case 1 (since $S_a \setminus S_b \subset V_G$), then it follows that

$$\begin{aligned}
 \tilde{\lambda}(\delta_{\tilde{G}}(S_a \setminus S_b)) &\geq 1 \\
 \implies \tilde{\lambda}(\delta_{\tilde{G}}(S)) &\geq 1.
 \end{aligned}$$

Thus, for this case the constraint in (8) is satisfied.

The above analysis shows that, given $\hat{\lambda}$ in the M-polytope of G , then $\tilde{\lambda}$, with the construction in (7), satisfies (8) – and hence it is in the PM-polytope of \tilde{G} .

APPENDIX D

PROOF OF THEOREM 4

We here prove Theorem 4 – which is a simplified version of the result in [11] – by following the logic in [14, Theorem 25.6]. In particular, the goal of the proof is to show that, for *any* odd cut U in the graph \tilde{G} , we can always consider any Gomory-Hu tree T of \tilde{G} , and find an odd cut W_{f^*} inside it, such that $\delta_{\tilde{G}}(U) \geq \delta_{\tilde{G}}(W_{f^*})$. Since this is true for all odd cuts, then it is also true for the minimum odd cut. This implies that one of the odd cuts in a Gomory-Hu tree is actually a minimum odd cut in \tilde{G} . In the remainder of this appendix, we formalize this argument.

Let $T = (V_{\tilde{G}}, F, \alpha_T)$ be a Gomory-Hu tree of the graph $\tilde{G} = (V_{\tilde{G}}, E_{\tilde{G}}, \lambda)$, where we assume that $|V_{\tilde{G}}|$ is even. Then, $\forall f \in F$, we let W_f be one of the two components of $T \setminus f$. Let U be an arbitrary odd cut in \tilde{G} . Applying U to the Gomory-Hu tree T gives us $\delta_T(U)$, which we refer to as the cut-set of U , i.e., the set of edges that separate U from U^c in the Gomory-Hu tree T . Note that $\delta_T(W_f) = f$. Thus, we have that

$$\begin{aligned}
 \delta_T(U) &= \bigtriangleup_{f \in \delta_T(U)} \{f\} \\
 &= \bigtriangleup_{f \in \delta_T(U)} \delta_T(W_f) \\
 &\stackrel{(a)}{=} \delta_T \left(\bigtriangleup_{f \in \delta_T(U)} W_f \right), \quad (17)
 \end{aligned}$$

where $A \Delta B$ is the symmetric difference between the sets A and B , and where in (a) we appeal to the property of cuts that for any two sets $S_1, S_2 \subset V_{\tilde{G}}$, we have that

$$\delta(S_1) \Delta \delta(S_2) = \delta(S_1 \Delta S_2).$$

Since cuts in a tree have unique cut-sets (excluding complements), then from (17), we have that

$$\bigtriangleup_{f \in \delta_T(U)} W_f = U \quad \text{or} \quad \bigtriangleup_{f \in \delta_T(U)} W_f = U^c.$$

Moreover, since $|V_{\tilde{G}}|$ is even and, by assumption, $|U|$ is odd, then $|U^c|$ is also odd. Thus, in any case, we have that $\left| \bigtriangleup_{f \in \delta_T(U)} W_f \right|$ is odd. Finally, the last step in the proof is to appeal to the following property that relates the cardinality of two sets to their symmetric difference. For any two sets A and B , we have that

$$|A \Delta B| = |A| + |B| - 2|A \cap B|. \quad (18)$$

Thus, if $|A \Delta B|$ is odd, then either $|A|$ or $|B|$ is odd. This extends to the case, when we have $\Delta_{i=1}^M A_i$. In this case, at least one of the sets A_i has an odd cardinality. Applying this observation to the set $\bigtriangleup_{f \in \delta_T(U)} W_f$, we have that for some $f^* \in \delta_T(U)$, $|W_{f^*}|$ is odd.

Let $f^* = (s, t)$ and recall that we assumed in our proof that the set U is an arbitrary odd cut in \tilde{G} . Moreover, the

edge f^* belongs to both $\delta_T(W_{f^*})$ and $\delta_T(U)$. Thus, both U and W_{f^*} represent an $s - t$ cut in \tilde{G} . From the definition of the Gomory-Hu tree (see Definition 1), W_{f^*} is a minimum cut for $s - t$. Thus, we have that

$$\tilde{\lambda}(\delta_{\tilde{G}}(U)) \geq \tilde{\lambda}(\delta_{\tilde{G}}(W_{f^*})).$$

Furthermore, since $|W_{f^*}|$ is odd, then we have proved that, for any odd cut U in \tilde{G} , we can find one cut W_{f^*} in T (cuts by removing one edge) that is odd and such that $\tilde{\lambda}(\delta_{\tilde{G}}(U)) \geq \tilde{\lambda}(\delta_{\tilde{G}}(W_{f^*}))$. This concludes the proof of Theorem 4.

REFERENCES

- [1] N. Alliance, “5G white paper,” *Next generation mobile networks, white paper*, 2015.
- [2] Y. H. Ezzeldin, M. Cardone, C. Fragouli, and G. Caire, “Gaussian 1-2-1 networks: Capacity results for mmwave communications,” in *IEEE International Symposium on Information Theory (ISIT)*, 2018.
- [3] F. Parvaresh and R. Etkin, “Efficient capacity computation and power optimization for relay networks,” *IEEE Transactions on Information Theory*, vol. 60, no. 3, pp. 1782–1792, 2014.
- [4] Y. H. Ezzeldin, M. Cardone, C. Fragouli, and D. Tuninetti, “Efficiently finding simple schedules in Gaussian half-duplex relay line networks,” in *IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 471–475.
- [5] 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, 2014.
- [6] 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, 2016.
- [7] 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.
- [8] J. Edmonds, “Maximum matching and a polyhedron with 0, 1-vertices,” *Journal of Research of the National Bureau of Standards B*, vol. 69, no. 125-130, pp. 55–56, 1965.
- [9] R. E. Gomory and T. C. Hu, “Multi-terminal network flows,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 4, pp. 551–570, 1961.
- [10] M. Grötschel, L. Lovász, and A. Schrijver, “The ellipsoid method and its consequences in combinatorial optimization,” *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981.
- [11] M. W. Padberg and M. R. Rao, “Odd minimum cut-sets and b-matchings,” *Mathematics of Operations Research*, vol. 7, no. 1, pp. 67–80, 1982.
- [12] I. Bárány, “A generalization of carathéodory’s theorem,” *Discrete Mathematics*, vol. 40, no. 2-3, pp. 141–152, 1982.
- [13] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*. Springer Science & Business Media, 2012, vol. 2.
- [14] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003, vol. 24.