An Algorithmic Framework for Wireless Information Flow

Michel X. Goemans
Department of Mathematics,
MIT,
Cambridge, USA.
Email: goemans@math.mit.edu

Satoru Iwata
RIMS,
Kyoto University,
Kyoto, Japan.
Email: iwata@kurims.kyoto-u.ac.jp

Rico Zenklusen
Institute for Operations Research,
ETH Zurich,
Zurich, Switzerland.
Email: rico.zenklusen@ifor.math.ethz.ch

Abstract—**We consider the wireless relay network model as introduced by Avestimehr, Diggavi and Tse [2] for approximating Gaussian relay channels and show that it is a special case of a more abstract flow model that we introduce in this paper. This flow model is based on linking systems, a combinatorial structure with a tight connection to matroids. A main advantage of this flow model is that properties and algorithms can easily be derived from existing theory on matroids and linking systems. In particular we show a max-flow min-cut theorem and submodularity of cuts. Furthermore, efficient algorithms for matroid intersection or for matroid partition can be used for finding a maximum flow and a minimum cut. Thus, this approach can profit from well-established matroid (intersection or partition) algorithms, leading to faster algorithms for large capacity networks. Another advantage of our approach is that it is easy to extend or adapt it to similar problems. In particular, the algorithm we present for finding maximum flows can easily be adapted to find a maximum flow with minimum costs when costs are introduced on the inputs and outputs of the relays.**

I. Introduction

We consider the deterministic wireless relay channel model as introduced by Avestimehr, Diggavi and Tse [2], which we call the ADT model. This model incorporates two key features of wireless relay networks, namely broadcasting and superposition and it can be used as a deterministic approximation to Gaussian relay channels. The signals are considered to be elements of a finite field and the interactions between the signals are assumed to be linear. A main advantage of the ADT model compared with the Gaussian model is its relatively simple structure, which makes it easier to analyze. In particular, Avestimehr, Diggavi and Tse have shown [3] a max-flow min-cut theorem for the ADT model. Recently, Amaudruz and Fragouli [1] have presented an efficient algorithm, based on augmenting paths, for finding a maximum flow in an ADT model, i.e., a relay encoding strategy that achieves the min-cut value. Both the max-flow min-cut result as well as the efficient algorithm for finding an encoding strategy are rather technical. In parallel to this work, Sadegh Tabatabaei Yazdi and Savari [9] developed another polynomial algorithm for finding a maximum flow in an ADT model, which is based on an extended Rado-Hall transversal theorem.

This work was partially supported by NSF contract CCF-0829878 and by ONR grant N00014-05-1-0148.

In this article we introduce a new flow network, called *linking network*, that generalizes the ADT model. Linking networks consist of combined *linking systems*, a notion introduced by Schrijver [10] in 1978 that has tight relations to matroids. An important advantage of reducing the ADT model to linking networks is that long-standing results on matroids and linking systems can be used to derive properties of linking networks and efficient algorithms, typically leading to clean results with simple proofs. In particular, we show a max-flow min-cut theorem and submodularity of the cut-value function; this property allows us to find a minimum cut by applying a standard algorithm for minimizing submodular functions. For finding a maximum flow in a linking network, we present two algorithms that reduce the problem to the matroid intersection and partition problems. These problems are well-studied in matroid theory and many efficient algorithms are known to solve them. The algorithm presented by Amaudruz and Fragouli [1] for finding a maximum flow in an ADT model as well as an algorithm of Schrijver [10], which finds a maximum flow in a linking network (Schrijver calls this the *cascading problem*) can be seen as particular ways of solving the matroid problems we present. The algorithms we derive are considerably faster than previous algorithms on large-capacity networks.

Using the linking network formulation of the ADT model allows us to easily extend and adapt the results to similar problems. In particular, the maximum flow algorithm can be adapted to find a maximum flow with minimum cost when costs are introduced on the inputs and outputs of the relays. The approach presented in this paper can also be extended to a capacitated network model by using polylinking systems instead of linking systems. To simplify the presentation we only consider the uncapacitated case where linking systems are sufficient.

The article is organized as follows. In Section II, we recall the ADT model. In Section III, the new flow model based on linking networks is presented and we show why the ADT model can be seen as special case of linking networks. In Section IV, properties and algorithms for linking networks are presented, including a max-flow min-cut theorem and efficient algorithms based on matroid theory for finding a maximum flow, a minimum cut and a minimum cost flow.

II. The ADT model

The ADT flow model was introduced in [2] as a linear deterministic approximation of Gaussian relay channels in a wireless setting. It takes into account broadcasting and interference effects. The random noise of a Gaussian channel is approximated by introducing between every pair of relays a threshold on the number of bits that can be transmitted from one relay to the other. A principal motivation for introducing the ADT flow model was to approximately determine the capacity of networked Gaussian channels. Whereas very little is known of how to compute the capacity of nontrivial combinations of Gaussian channels, in the ADT model the capacity can be determined in polynomial time.

Given is a collection of *relays* $\mathcal{N} = \{N^1, \dots, N^q\}$, also called *nodes*, each of which is a set of $2b$ vertices for some global constant $b \in \mathbb{N}$. For each node $N \in \mathcal{N}$, its vertices are partitioned into two groups of b vertices, called *outputs* and *inputs*. For a node $N^i \in \mathcal{N}$, we denote by $\{v_1^i, \dots, v_b^i\}$ its inputs and by $\{w_1^i, \dots, w_b^i\}$ its outputs. We call the set of all vertices V, the set of all outputs O and the set of all inputs I. The nodes are partitioned into layers $T_1, \dots, T_r$, where $T_1 = \{N^1\}$ and $T_r = \{N^q\}$. The node N^1 is called *sender* and N^q is called *receiver*. The goal is to send signals in some finite field $\mathbb{F}_p$ from the sender to the receiver. Every input can send a signal to outputs of nodes in the next layer. How the signals are sent from the inputs of nodes in one layer to the outputs of the nodes in the next layer is described by a set of arcs $A \subseteq I \times O$ given as follows. For every pair of nodes N^i, N^j such that N^i is in the layer immediately preceding the layer containing N^j, a number $n_{ij} \in \{0, \dots, b\}$ is given and the arcs of A connecting inputs of N^i to outputs of N^j are given by $\{(v_1^i, w_{b+1-n_{ij}}^j), \dots, (v_{n_{ij}}^i, w_b^j)\}$. If $n_{ij} = 0$ then there are no arcs connecting inputs of N^i to outputs of N^j. The constants n_{ij} represent the number of bits that relay N^j can receive from relay N^i. Thus, a small value for n_{ij} models that signals sent from relay N^i to N^j suffer a high noise. We call the triple $G = (V, A, \mathcal{N})$ an *ADT network*. See Figure 1 for an illustration of an ADT network.

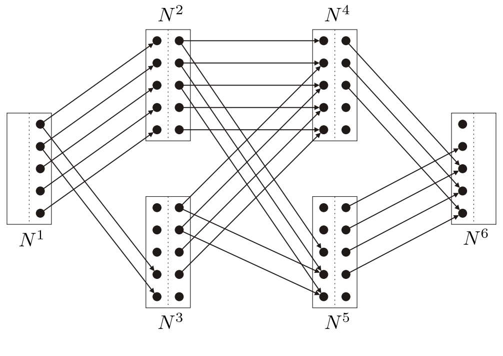

Figure 1: Example of an ADT network with $q = 6$. The rectangles represent nodes. The vertices contained in the left half of each node are outputs and the ones in the right half are inputs. In this example we have $n_{12} = 5, n_{13} = 2, n_{24} = 5, n_{25} = 3, n_{34} = 4, n_{35} = 2, n_{46} = 3, n_{56} = 4$.

We denote by I_i the inputs in layer i and O_i the outputs in layer i, i.e., $I_i = I \cap (\cup_{N \in T_i} N)$, $O_i = O \cap (\cup_{N \in T_i} N)$. See Figure 2 for an illustration.

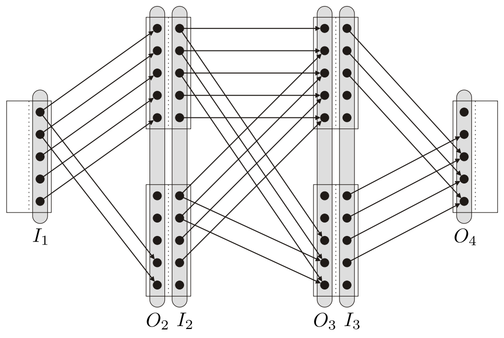

Figure 2: Representation of the output sets O_i and input sets I_i.

Signals are sent in the following way. At each input w_l^1 of the sender, a signal $s \in \mathbb{F}_p$ can be broadcast to the next layer. The signal is sent to all outputs in the next layer that are connected by an arc to w_l^1. This models broadcasting, i.e., a signal in the wireless network cannot be directed towards a single particular output but may reach other outputs as a side effect. If an output receives signals from different inputs, interference between the received signals happens which is modeled as follows. The output receives the sum in $\mathbb{F}_p$ of the incoming signals. Every node receiving signals at its outputs can resend them over its inputs. A signal received at a particular output can be resent over any input of the same node. However, every input can be used at most once. Not every output has to be linked to an input and thus not every input has to be used. The assignment of inputs of a given node to its outputs is called *wiring*. Finally, the receiver receives at its outputs a set of linear combinations of the signals sent by the sender. The signals have to be sent in such a way that the receiver can decode the original signals, i.e., if k signals are sent from the sender, then to properly decode the signals, the receiver needs to receive k signals that are linearly independent combinations (over $\mathbb{F}_p$) of the k signals sent by the sender. The task is to send the largest number of decodable signals from the sender to the receiver. As observed in [2], decodability of the signal does not depend on the exact wiring inside the nodes but only on the set of outputs and inputs that are used inside the nodes. Since the exact wiring is not important, a flow in an ADT network can be defined as the set of vertices used for receiving or sending signals as follows. For some set $U \subseteq V$, we denote by $M[U] \in \mathbb{F}_p^{(U \cap I) \times (U \cap O)}$ the matrix such that for $i \in U \cap I$ and $j \in U \cap O$, $M[U]_{ij} = 1$ if $(i, j) \in A$ and $M[U]_{ij} = 0$ otherwise. A *flow* F in the ADT network $G = (V, A, N)$ is a subset of the vertices V such that:

i) For every node $N \in \mathcal{N}$, $|F \cap N \cap O| = |F \cap N \cap I|$, i.e., the number of used outputs in every node equals the number of used inputs.
ii) The matrix $M[F]$ has full rank (in $\mathbb{F}_p$).

The first condition makes sure that for every node $N \in \mathcal{N}$, it is possible to wire the signals received at its outputs $N \cap O \cap F$ to the inputs $N \cap I \cap F$. The second condition guarantees that the signals received at the outputs $F \cap N^q$ of the receiver are linearly independent combinations of the signals sent by the sender and thus are decodable. Since the network is layered, the second condition can also be restated as a condition that has to hold for every pair of consecutive layers.

ii') For every $i = 1, \cdots, r-1$, $M[F \cap (I_i \cup O_{i+1})]$ has full rank.

The value of a flow F is measured by $|F \cap N^1| = |F \cap N^q|$. For finding a flow of maximum value in the ADT model, Amaudruz and Fragouli give a combinatorial algorithm. The running time is bounded by $O(n|A|(\phi^*)^5)$, where ϕ^* is the capacity of the network, i.e, the maximum flow value [1, Proposition 3.3].

In [2] the following notion of a source-destination cut was introduced, which we call an *ADT cut*. An *ADT cut* is a set $C \subseteq V$ such that for every node $N \in \mathcal{N}$, either $N \subseteq C$ or $N \cap C = \emptyset$, and $N^1 \subseteq C$, $N^q \cap C = \emptyset$. With every ADT cut C a value is associated which is given by $\sum_{i=1}^{r-1} \operatorname{rank}(M[(C \cap I_i) \cup (O_{i+1} \setminus C)])$. It is relatively easy to observe that the value of any ADT cut is an upper bound on the value of a maximum ADT flow. Furthermore, Avestimehr et al. [3] show that the value of a minimum ADT cut equals the value of a maximum ADT flow.

III. A NEW FLOW MODEL BASED ON LINKING SYSTEMS

Before introducing the new flow model based on linking systems, we recall some results about matroids and linking systems. For proofs and further information, see [12] for matroids and [10], [11] for linking systems.

A. Preliminaries on matroids and linking systems

For $k \in \mathbb{N}$, we use the notation $[k] = \{1, \ldots, k\}$. Let S be a finite set. For a subset $A \subseteq S$ and $x \in \mathbb{R}^S$, we denote by $x(A)$ the sum of the components of x corresponding to A, i.e., $x(A) = \sum_{a \in A} x(a)$. A function f from the subsets of S to the reals is called *submodular* if $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ for all $A, B \subseteq S$.

Matroids: A matroid is a pair $M = (S, \mathcal{I})$ where S is a finite ground set and $\mathcal{I} \subseteq 2^S$ is a nonempty family of subsets of S which are called *independent sets* and have to satisfy the following conditions:

i) If $I \in \mathcal{I}$ and $J \subseteq I$ then $J \in \mathcal{I}$,
ii) if $I, J \in \mathcal{I}$ with $|I| > |J|$ then $\exists z \in I \setminus J$ with $J \cup \{z\} \in \mathcal{I}$.

The *bases* of a matroid are its maximal independent sets and it is well-known that all bases of a matroid have the same cardinality. To every matroid $M = (S, \mathcal{I})$, a *rank function* $\rho : 2^S \to \mathbb{Z}_+$ is associated; it is defined by $\rho(I) = \max\{|J| \mid J \subseteq I, J \in \mathcal{I}\}$. The rank function completely describes the underlying matroid since a set $I \subseteq S$ is independent if and only if $|I| = \rho(I)$. The rank function of a matroid can be shown to be submodular. An important algorithmic result in matroid theory is that the greedy algorithm allows to find an independent set of a matroid that is maximum with respect to some given weight $c : S \to \mathbb{R}$. Furthermore, for two matroids $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ that share the same ground set, there are efficient algorithms for finding a set of maximum weight that is independent in both matroids; this is the weighted matroid intersection problem. Similarly, there are efficient algorithms to find a maximum weight common base of two matroids that share the same ground set, respectively to determine that no common base exists.

Linking systems and linking functions: Linking systems were introduced by Schrijver [10]. They are functions that map subsets of one finite ground set to subsets of another finite ground set while preserving some useful structures. In this work, linking systems are used to describe how signals can be sent from one layer of inputs to the next layer of outputs and for the wiring inside nodes. Two key examples of linking systems are given just after the formal definition below.

A *linking system* is a triple (V_1, V_2, Λ), where V_1 and V_2 are finite sets, called *ground sets*, and $\Lambda \subseteq 2^{V_1} \times 2^{V_2}$ such that the following conditions are satisfied:

i) If $(P_1, P_2) \in \Lambda$, then $|P_1| = |P_2|$,
ii) if $(P_1, P_2) \in \Lambda$ and $Q_1 \subseteq P_1$, then $\exists Q_2 \subseteq P_2$ with $(Q_1, Q_2) \in \Lambda$,
iii) if $(P_1, P_2) \in \Lambda$ and $Q_2 \subseteq P_2$, then $\exists Q_1 \subseteq P_1$ with $(Q_1, Q_2) \in \Lambda$,
iv) if $(P_1, P_2), (Q_1, Q_2) \in \Lambda$, then $\exists (R_1, R_2) \in \Lambda$ such that $P_1 \subseteq R_1 \subseteq P_1 \cup Q_1$ and $Q_2 \subseteq R_2 \subseteq P_2 \cup Q_2$.

If two sets $P_1 \subseteq V_1$ and $P_2 \subseteq V_2$ satisfy $(P_1, P_2) \in \Lambda$ then we say that P_1 is linked to P_2 (by Λ). In the following we give two examples of linking systems, which we later use to model ADT flows. For proofs and further information, see [10].

Example 1 (Linking systems induced by bipartite graphs). Let $G = (V_1, V_2, E)$ be an undirected bipartite graph and let $\Lambda \subseteq 2^{V_1} \times 2^{V_2}$ be such that $(P_1, P_2) \in \Lambda$ if and only if there exists a matching M in G such that the set of vertices that are adjacent to edges in M is $P_1 \cup P_2$. Then (V_1, V_2, Λ) is a linking system.

Example 2 (Linking systems induced by matrices). Let F be a field and $A \in F^{V_1 \times V_2}$ be a matrix over the field F where V_1 and V_2 are two finite sets representing the rows and columns of A. Let $\Lambda \subseteq 2^{V_1} \times 2^{V_2}$ such that $(P_1, P_2) \in \Lambda$ if and only if the submatrix of A generated by the rows P_1 and columns P_2 has full rank. Then (V_1, V_2, Λ) is a linking system.

Notice that the definition of a linking system is symmetric with respect to the two ground sets. For a linking system (V_1, V_2, Λ) we denote by $(V_2, V_1, \overline{\Lambda})$ the linking system defined by $\overline{\Lambda} = \{(P_2, P_1) \mid (P_1, P_2) \in \Lambda\}$.

To every linking system (V_1, V_2, Λ), a corresponding *linking function* $\lambda : 2^{V_1} \times 2^{V_2} \to \mathbb{R}_+$ is associated that is defined by

$$\lambda(P_1, P_2) = \max\{|Q_1| \mid Q_1 \subseteq P_1, Q_2 \subseteq P_2, (Q_1, Q_2) \in \Lambda\}.$$

Thus, $\lambda(P_1, P_2)$ is the maximum size of an element lying in P_1 that can be linked to an element lying in P_2. A linking function completely describes the underlying linking system because two sets $P_1 \subseteq V_1, P_2 \subseteq V_2$ are linked if and only if $|P_1| = |P_2|$ and $\lambda(P_1, P_2) = |P_1|$. In this article, we assume that the linking function can be evaluated in polynomial time. This is the case for all linking systems used in this paper. The following proposition gives a characterization of linking functions. This proposition as well as those following in this section are proved in Schrijver [10].

Proposition 1. *Let V_1, V_2 be two finite sets and let $\lambda : 2^{V_1} \times 2^{V_2} \to \mathbb{Z}_+$. Then λ is the linking function of a linking system on ground sets V_1 and V_2 if and only if it satisfies the following conditions for all $P_1, Q_1 \subseteq V_1$ and $P_2, Q_2 \subseteq V_2$.*

i) $\lambda(P_1, P_2) \leq \min\{|P_1|, |P_2|\}$,

ii) *if $Q_1 \subseteq P_1$ and $Q_2 \subseteq P_2$ then $\lambda(Q_1, Q_2) \leq \lambda(P_1, P_2)$,*

iii) $\lambda(P_1 \cap Q_1, P_2 \cup Q_2) + \lambda(P_1 \cup Q_1, P_2 \cap Q_2) \leq \lambda(P_1, P_2) + \lambda(Q_1, Q_2)$.

The third condition in the above proposition is called *bisubmodularity*. The following proposition shows that a linking system can be seen as a particular way of representing a matroid. This tight link between linking systems and matroids allows to deduce many results for linking systems from results in matroid theory.

Proposition 2. *Let (V_1, V_2, Λ) be a linking system with disjoint ground sets. Then the set $\mathcal{B}_\Lambda = \{P_1 \cup (V_2 \setminus P_2) \mid (P_1, P_2) \in \Lambda\}$ forms the set of bases of a matroid on the ground set $V_1 \cup V_2$.*

For a linking system (V_1, V_2, Λ) with disjoint ground sets we denote by M_Λ the matroid corresponding to the bases $\mathcal{B}_\Lambda$, and by ρ_Λ the rank function of M_Λ. The following proposition shows the relation between the rank function ρ_Λ and the linking function λ.

Proposition 3. *Let (V_1, V_2, Λ) be a linking system with disjoint ground sets and with linking function λ. For $P_1 \subseteq V_1$ and $P_2 \subseteq V_2$,*

$$\rho_\Lambda(P_1 \cup P_2) = \lambda(P_1, V_2 \setminus P_2) + |P_2|.$$

Combining linking systems: Two linking systems that share one of the ground sets can be chained together to obtain a new linking system. This operation is called the *product* of linking systems and is defined as follows.

Proposition 4. *Let (V_1, V_2, Λ_1) and (V_2, V_3, Λ_2) be two linking systems, with linking functions λ_1 and λ_2 and define*

$$\Lambda_1 \star \Lambda_2 = \{(P_1, P_3) \in 2^{V_1} \times 2^{V_3} \mid \exists P_2 \subseteq V_2 \text{ with } (P_1, P_2) \in \Lambda_1, (P_2, P_3) \in \Lambda_2\}.$$

Then $(V_1, V_3, \Lambda_1 \star \Lambda_2)$ is again a linking system with linking function

$$(\lambda_1 \star \lambda_2)(P_1, P_3) = \min_{P_2 \subseteq V_2} \{\lambda_1(P_1, P_2) + \lambda_2(V_2 \setminus P_2, P_3)\}.$$

B. Linking networks

Let $r \geq 2$ be an integer, $V_1, \dots, V_r$ be finite disjoint sets and let $(V_i, V_{i+1}, \Lambda_i)$ be a linking system with linking function λ_i for $i \in [r-1]$. We call the tuple $G = (V, \Lambda)$, where $V = (V_1, \dots, V_r)$ and $\Lambda = (\Lambda_1, \dots, \Lambda_{r-1})$ a *linking network*. The sets $V_1, \dots, V_r$ are called the *layers* of the network G and elements of these sets are called *vertices*. Furthermore, V_1 is the *source layer* and its vertices are called *sources*. Similarly, V_r is the *destination layer* and its vertices are called *destinations* or *sinks*. A *flow in a linking network*, or also simply called *flow*, is a tuple $F = (F_1, \dots, F_r)$ with $F_i \subseteq V_i$ for $i \in [r]$ and satisfying $(F_i, F_{i+1}) \in \Lambda_i$ for $i \in [r-1]$. The value of a flow F is defined by $\nu(F) = |F_1|$. Notice that by definition of a flow we have $|F_1| = |F_i|$ for $i \in [r]$. A *$V_1 - V_r$ cut in a linking network*, or also called *source-destination cut* or simply *cut*, is a tuple $C = (C_1, \dots, C_r)$ such that $C_i \subseteq V_i \ \forall i \in [r]$, $C_1 = V_1$ and $C_r = \emptyset$. The value of a cut C is defined by $\phi(C) = \sum_{i=1}^{r-1} \lambda_i(C_i, V_{i+1} \setminus C_{i+1})$. If there is no danger of ambiguity we also represent a cut $C = (C_1, \dots, C_r)$ by the set $\cup_{i=1}^{r} C_i$ and use the cut function ϕ also for this cut representation.

C. Reducing the ADT model to a linking network

Consider an ADT network as introduced in Section II with q layers of nodes. The ADT model is represented by a linking network defined over the $2q-2$ layers $I_1, O_2, I_2, \dots, O_q$ using two types of linking systems. For $i \in [q-1]$, the linking system that links I_i to O_{i+1} is the linking system induced by the matrix $M[I_i \cup O_{i+1}]$. This ensures that a set of linear independent signals that are sent from one layer of inputs to the next layer of outputs results in linear independent signals at the outputs. Thus the condition ii) of an ADT flow is satisfied. For $i \in \{2, \dots, q\}$, the set O_i is linked to I_i by a linking system induced by the bipartite graph that contains an edge between a vertex $v \in O_i$ and $w \in I_i$ if v and w are in the same node. These linking systems enforce that a linking flow satisfies property i) of an ADT flow. Hence a linking flow in this linking network is indeed an ADT flow in the corresponding ADT model and vice versa.

However, in the thus described linking network used for representing an ADT network, the notion of cut is slightly different to the notion of an ADT cut. An ADT cut satisfies that each node is either included in the cut or excluded, whereas the definition of a cut in a linking network allows to contain part of a node. However, we show below that, from any minimum cut in the linking network, one can easily deduce a corresponding ADT cut of no greater value (and hence of minimum value as well). This implies that the value of a minimum ADT cut is equal to the value of a minimum cut in the corresponding linking network.

More precisely, let $G = (V, \Lambda)$ be a linking network with r layers that stems from an ADT model. Let $C = (C_1, \dots C_r)$ be a minimum cut in G. Assume there is a relay N with outputs O_N in layer i and inputs I_N in layer $i+1$ such that $(C_i \cap C_{i+1}) \cap N$ is a proper subset of N. We show that if $|C_i \cap O_N| \leq |I_N \setminus C_{i+1}|$ then excluding the vertices from node N from the cut results again in a minimum cut, otherwise the vertices from node N can be included from C to obtain another minimum cut. Repeating this argument for every relay N leads to an ADT cut of minimum value as well.

Assume $|C_i \cap O_N| \leq |I_N \setminus C_{i+1}|$ (the other case is analogue). Thus, $\lambda_i(C_i \cap O_N, I_N \setminus C_{i+1}) = |C_i \cap O_N|$. Consider the cut

$C' = (C_1, \ldots, C_{i-1}, C'_i, C'_{i+1}, C_{i+2}, \ldots, C_r)$ where $C'_i = C_i \backslash N$ and $C'_{i+1} = C_{i+1} \backslash N$. Note that Λ_i is a linking system induced by a bipartite graph containing an edge between any output-input pair within any node consisting of vertices in $O_i \cup I_{i+1}$. Hence,

$$\lambda_i(C_i, V_{i+1} \setminus C_{i+1}) = \sum_{\substack{N \in \mathcal{N}, \\ N \subseteq O_i \cup I_{i+1}}} \lambda_i(C_i \cap N, N \setminus C_{i+1}),$$

which implies

$$\begin{aligned}\lambda_i(C'_i, V_{i+1} \setminus C'_{i+1}) - \lambda_i(C_i, V_{i+1} \setminus C_{i+1}) \\ = -\lambda_i(C_i \cap N, N \setminus C_{i+1}) = -|C_i \cap O_N|. \quad (1)\end{aligned}$$

Furthermore, we have

$$\lambda_{i-1}(C'_{i-1}, V_i \setminus C'_i) - \lambda_{i-1}(C_{i-1}, V_i \setminus C_i) \leq |C_i \cap O_N|, \quad (2)$$

since C'_i is obtained from C_i by removing $|C_i \cap O_N|$ elements and each element that is removed can decrease the contribution of λ_{i-1} to the value of the cut by at most 1. Additionally, since $C_{i+1} \subseteq C'_{i+1}$, we get by monotonicity of the linking function

$$\lambda_{i+1}(C'_{i+1}, V_{i+2} \setminus C'_{i+2}) \leq \lambda_{i+1}(C_{i+1}, V_{i+2} \setminus C_{i+2}). \quad (3)$$

Combining (1), (2) and (3) we get $\phi(C') \leq \phi(C)$.

IV. Results and algorithms on linking networks

In the following we deduce properties and algorithms for linking networks, and thus also for the ADT model, from the facts on matroids and linking systems that were presented in the previous section.

A. Duality theorem and submodularity of cuts

Theorem 5. *In any linking network, the value of a maximum flow is equal to the value of a minimum cut.*

Proof: Let $G = (V, \Lambda)$ be a linking network with r layers and we define $\widetilde{\Lambda} = \Lambda_1 \star \cdots \star \Lambda_{r-1}$, which is a linking system by Proposition 4. By the definition of the product of linking systems we have that there is a linking flow in G of some given value $q \in \mathbb{Z}_+$ if and only if there exists a pair $(P_1, P_r) \in \widetilde{\Lambda}$ with $|P_1| = q$. Thus, the value of a maximum linking flow in G is equal to $\widetilde{\lambda}(V_1, V_r)$, where $\widetilde{\lambda}$ is the linking function corresponding to $\widetilde{\Lambda}$. By Proposition 4 and the definition of the value of a cut we have

$$\begin{aligned}\widetilde{\lambda}(V_1, V_r) &= \min \Big\{ \lambda_1(V_1, V_2 \backslash P_2) + \sum_{i=2}^{r-2} \lambda_i(P_i, V_{i+1} \backslash P_{i+1}) \\ &\qquad + \lambda_r(P_{r-1}, V_r) \mid P_2 \subseteq V_2, \ldots, P_{r-1} \subseteq V_{r-1} \Big\} \\ &= \min \Big\{ \phi\Big(V_1 \cup \bigcup_{i=2}^{r-1} P_i\Big) \mid P_2 \subseteq V_2, \ldots, P_{r-1} \subseteq V_{r-1} \Big\},\end{aligned}$$

which is the value of a minimum cut in G. ■

Theorem 6. *The function ϕ that associates to every cut its value is submodular.*

Proof: Let $G = (V, \Lambda)$ be a linking network with r layers and let $C = (C_1, \ldots, C_r)$ be a cut in G. By definition the value of the cut C is given by $\phi(C) = \sum_{i=1}^{r-1} \lambda_i(C_i, V_{i+1} \setminus C_{i+1})$. By the bisubmodularity property of linking functions, we have that for $i \in [r-1]$ the function g_i defined on $2^{V_i} \times 2^{V_{i+1}}$ by $g_i(P_i, P_{i+1}) = \lambda_i(P_i, V_{i+1} \setminus P_{i+1})$ is submodular. Hence, $\phi(C)$ is the sum of submodular functions, and thus submodular. ■

The above submodularity result has important algorithmic implications since there exist polynomial algorithms for minimizing submodular functions. A minimum cut in a linking network can then be obtained by minimizing the submodular cut-value function in $O((m^4\alpha + m^5)\log(m))$ time, where m the total number of vertices in the linking network and α is the time needed to evaluate the value of a given cut [7], [8]. An efficient algorithm for finding a minimum cut can easily be transformed into an efficient algorithm for finding a maximum flow as follows. Consider the vertices in $\cup_{i=1}^r V_i$ in any order. If removing a vertex from the corresponding linking system does not decrease the value of a minimum cut, we remove the vertex from the graph. One can easily check that the remaining vertices form a maximum flow. In the following we present a much more efficient and direct algorithm for obtaining a maximum flow in a linking system. A minimum cut is obtained as a by-product of the algorithm. Even if one is just interested in determining the minimum cut of a linking network, the algorithm to be presented is considerably faster than a general submodular function minimization algorithm as mentioned above.

B. Optimizing in linking networks

Here we show how standard matroid algorithms can be used to efficiently find a maximum flow in a linking network. We show that one can either use algorithms for matroid intersection or for matroid partition.

Recall that, for any single linking system (V_1, V_2, Λ), two matroids can be defined, M_Λ and $M_{\overline{\Lambda}}$; for any $(P_1, P_2) \in \Lambda$, $P_1 \cup (V_2 \backslash P_2)$ is a base of the first matroid while $(V_1 \backslash P_1) \cup P_2$ is a base of the second matroid.

Matroid Intersection: Let $G = (V, \Lambda)$ be a linking network with r layers. For simplicity, we first consider the case in which $r = 2k$ is even; this is the case for linking networks arising from the ADT model. We construct two matroids M_o (for odd) and M_e (for even). Both have $\cup_{i=1}^r V_i$ as ground set. M_o is defined as the disjoint union $M_{\Lambda_1} \oplus M_{\Lambda_3} \oplus \cdots \oplus M_{\Lambda_{2k-1}}$ of the matroids corresponding to the odd-numbered linking systems; a set $I \subseteq \cup_{i=1}^r V_i$ is independent in matroid M_o if for any odd integer j with $j \in [r-1]$, we have $I \cap (V_j \cup V_{j+1})$ is independent in M_{Λ_j}. Thus, for any flow $F = (F_1, \cdots, F_r)$ in G, we have that $m(F) := F_1 \cup (V_2 \setminus F_2) \cup F_3 \cup \cdots \cup (V_{2k} \setminus F_{2k})$ is a base of M_o. In order to define M_e, we first define a dummy *free* linking system (V_r, V_1, Λ_0) linking the last layer to the first, and having all possible pairs (P_r, P_1) with $|P_r| = |P_1|$, $P_r \subseteq V_r$ and $P_1 \subseteq V_1$ linked in Λ_0. We define M_e as the disjoint union $M_{\overline{\Lambda_0}} \oplus M_{\overline{\Lambda_2}} \oplus \cdots \oplus M_{\overline{\Lambda_{2k-2}}}$. Observe that for any flow F, we also have that $m(F)$ is a base of M_e; thus, $m(F)$ is a common base to M_e and M_o. Conversely, consider any common base B to M_e and M_o; by construction of M_e and M_o, we can derive a flow F from

B by setting

$$F_i = \begin{cases} B \cap V_i & 1 \le i < r, i \text{ odd} \\ V_i \setminus B & 1 < i \le r, i \text{ even.} \end{cases}$$

So far, we have assumed that the number r of layers in the linking network G is even. The case of an odd number of layers can be treated in several ways. One possibility is to add a final layer V_{r+1} and a free linking system $(V_r, V_{r+1}, \Lambda_r)$ (i.e. one for which any pair of sets of $V_r \times V_{r+1}$ of the same cardinality is linked). This gives a linking network G' with an even number of layers, and there is a trivial correspondence between flows in G and flows in G'.

In the case of an even number of layers, the correspondance between flows in the linking network G and common bases to M_e and M_o allows us to find a maximum flow in G by finding a common base B to M_e and M_o that maximizes $|B \cap V_1|$. This is a particular case of weighted matroid intersection, where the goal is to find, for every integer k, a set I of cardinality k independent for two matroids and which maximizes $\sum_{e \in I} w(e)$ for some weight vector w. In summary, the maximum flow in a linking network G can be found efficiently using any algorithm for weighted matroid intersection, see [12, Section 41.3].

For example, if we use the simple implementation of the algorithm of Frank [6], as described in [12, Section 41.3a], we can derive a maximum flow in time $O(nr\tau)$, where n is the maximum number of elements per layer, and τ is the time for constructing a certain auxiliary graph with respect to a common independent set I. For the ADT model described before, τ can be chosen to be $O(rn^3)$, leading to an overall running time of $O(r^2n^4)$. With the aid of the scaling algorithm in [13], one can improve this running time to $O(r^{1.5}n^{3.5}\log(nr))$.

The use of weighted matroid intersection to find a flow of maximum cardinality in a linking network $G = (V, \Lambda)$ straightforwardly generalizes to the problem of finding a maximum flow of minimum cost. In this extension, one is given a cost $c(v)$ for every $v \in \cup_{i=1}^r V_i$ and one would like to find a maximum flow $F = (F_1, F_2, \cdots, F_r)$ minimizing $\sum_{i=1}^r \sum_{v \in F_i} c(v)$.

Matroid partition: We now show another reduction allowing to solve linking flow problems using any matroid partition algorithm. Given any linking network $G = (V, \Lambda)$ with r layers, consider the matroid M^* defined as the union of the matroids M_{Λ_i} for $i \in [r-1]$. This is a matroid whose ground set $\cup_{i=1}^r V_i$ is the union of the ground sets of the M_{Λ_i}'s, and whose independent sets are $\{\cup_{i=1}^{r-1} I_i : I_i \in \mathcal{I}(M_{\Lambda_i})\}$ where $\mathcal{I}(M)$ denotes the family of independent sets of matroid M. The resulting union M^* is indeed a matroid, see [12, Chapter 42] for a proof and discussion. The independent sets of $\mathcal{I}(M^*)$ are called *partitionable*.

The connection between the maximum flow problem in G and independent sets in M^* is highlighted in the following theorem.

Theorem 7. *Let M^* be the matroid described above corresponding to a linking network $G = (V, \Lambda)$. Then*

1) *Given a flow $F = (F_1, \cdots, F_r)$ in G, we have that*

$$F_1 \cup \left(\cup_{i=2}^{r-1} V_i\right) \cup (V_r \setminus F_r) \in \mathcal{I}(M^*). \tag{4}$$

2) *Given any base B of M^* such that*

$$B \supseteq \cup_{i=2}^{r-1} V_i, \tag{5}$$

and expressed as $B = \cup_{i=1}^{r-1} I_i$ with $I_i \in \mathcal{I}(M_{\Lambda_i})$ one can derive a flow of G by letting:

$$F_i = \begin{cases} I_i \cap V_i & i \in [r-1] \\ I_r \setminus B & i = r. \end{cases} \tag{6}$$

In particular, $F_1 = B \cap V_1$.

Proof: 1) Given the flow $F = (F_1, \cdots, F_r)$, we know that $F_i \cup (V_{i+1} \setminus F_{i+1}) \in \mathcal{I}(M_{\Lambda_i})$ for $i \in [r-1]$, and therefore (4) holds.

2) Consider a base B of M^* with the required properties. We claim that each I_i is a base of M_{Λ_i}. Since all bases of a matroid have the same cardinality, this can be verified by showing that there is one basis of M^* that is a disjoint union of bases of M_{Λ_i}, for $i \in [r-1]$. The set $\cup_{i=2}^r V_i \in \mathcal{I}(M^*)$ is such a set since V_{i+1} is a basis of M_{Λ_i} for $i \in [r-1]$. Defining F by (6), we immediately get that $F = (F_1, \cdots, F_r)$ satisfies the definition of a flow. Furthermore, $F_1 = I_1 \cap V_1 = B \cap V_1$. ■

In M^* (as in any (non-disjoint) union of matroids), it is not completely straightforward to test independence of a set. This is the purpose of *matroid partition* algorithms which proceed incrementally. Given disjoint sets $I_i \in \mathcal{I}(M_{\Lambda_i})$, $I = \cup_{i=1}^{r-1} I_i$ and given $v \notin I$, one fundamental step of a matroid partition algorithm decides whether $I \cup \{v\} \in \mathcal{I}(M^*)$ and if so, finds disjoint $I'_i \in \mathcal{I}(M_{\Lambda_i})$ with $I \cup \{v\} = \cup_{i=1}^{r-1} I'_i$. This can be used to find a maximum flow in a linking network. Indeed, using Theorem 7, we can start from $I = \cup_{i=1}^{r-2} V_{i+1} \in \mathcal{I}(M^*)$ (I consists of all elements except the first and last layers) and repeatedly first check whether each element of V_1 can be added to our current set $I \in \mathcal{I}(M^*)$, and then do the same for V_r. This results in a base of M^* satisfying (5) and having as many elements of V_1 as possible; therefore, as in the proof of the theorem, we can extract a maximum flow. Observe that, for our initial set I, we have a trivial decomposition of it into $\cup_{i=1}^{r-1} I_i$ by taking $I_i = V_{i+1}$ for $i \in [r-2]$ and $I_{r-1} = \emptyset$.

We need now to discuss how the fundamental step in a matroid partition algorithm can be performed. There exist classical algorithms for the fundamental step, see [12, Section 42.3]. This boils down to constructing a digraph on $\cup_{i=1}^r V_i$ and finding a shortest directed path in it. The arc set of this digraph is $\cup_{i=1}^{r-1} E_i$ with

$$\begin{aligned} E_i = \{(s,t) \mid s \notin I_i,\, t \in I_i,\, I_i \cup \{s\} \setminus \{t\} \in \mathcal{I}(M_{\Lambda_i})\} \\ \subset (V_i \cup V_{i+1}) \times (V_i \cup V_{i+1}). \end{aligned}$$

For each $i \in [r-1]$, consider a vertex subset $S_i = \{s \in (V_i \cup V_{i+1}) \setminus I_i \mid I_i \cup \{s\} \in \mathcal{I}(M_{\Lambda_i})\}$. The fundamental step searches for a directed path from $v \notin I$ to any vertex in $\cup_{i=1}^{r-1} S_i$ with minimum number of arcs. If such a directed path exists, then $I \cup \{v\} \in \mathcal{I}(M^*)$, and exchanging the elements along the directed path provides a partition of $I \cup \{v\}$ into disjoint $I'_i \in \mathcal{I}(M_{\Lambda_i})$. Otherwise, $I \cup \{v\} \notin \mathcal{I}(M^*)$. Since, for every $i \in [r-2]$, we start from a base $I_i = V_i$ of M_{Λ_i}, we have that $S_i = \emptyset$ for $i \in [r-2]$; only S_{r-1} is non-empty. For the ADT model, constructing this digraph takes $O(rn^3)$, and this dominates the time to find appropriate directed path in it. As we are performing at most $2n$ fundamental steps, this gives an overall running time of $O(rn^4)$. Using the analysis technique by Cunningham [5], one can show that the total length of the directed paths in the above matroid partition algorithm is $O(rn \log n)$, which leads to an improved running time bound $O(rn^3 \log n)$.

Thus, for ADT networks with large capacity, for example $\phi^* = \Theta(n)$, both algorithms we presented (based on matroid intersection and matroid union) are considerably faster than the algorithm of Amaudruz and Fragouli [1].

Finding a minimum source-destination cut: We now show how a minimum source-destination cut can be obtained as a by-product of the above matroid partition algorithm (a similar argument works for the matroid intersection approach).

At the end of the algorithm we get a solution set $B = \cup_{i=1}^{r-1} I_i$ with $I_i \in \mathcal{I}(M_{\Lambda_i})$ which is a base of M^* and corresponds to a maximum flow by Theorem 7. We now consider the digraph for the fundamental step of the matroid partition algorithm that corresponds to this final solution. We know that this digraph has no directed path from $V_1 \setminus I_1$ to V_r; indeed such a path from $s \in V_1 \setminus I_1$ to $t \in V_r$ would mean that $B \setminus \{t\} \cup \{s\}$ is also a base of M^* implying that the algorithm should have added s when it considered it. In fact, since the vertices in $V_1 \cap I_1$ have no outgoing arcs (see the definition of E_1), there are no directed paths from V_1 to V_r. Let $W \subset \cup_{i=1}^{r-1} V_i$ be the set of all vertices that are reachable from the vertices V_1 in this digraph; thus $W \cap V_r = \emptyset$ and $V_1 \subseteq W$.

Theorem 8. *The set W is a minimum source-destination cut.*

Proof: Again let M^* be the union of the matroids M_{Λ_i} for $i \in [r-1]$ and let M^*_- be the restriction of the matroid M^* to the ground set $\cup_{i=1}^{r-2} V_i$, i.e., a set $I \subseteq \cup_{i=1}^{r-2} V_i$ is independent in M^*_- if it is independent in M^*. Since we considered vertices in V_1 prior to those in V_r, $B \setminus V_r$ is a base of M^*_-. Observe that W also corresponds to the set of vertices reachable from V_1 in the digraph for the fundamental step of the matroid partition algorithm corresponding to the matroid M^*_- . It is well-known in matroid theory (see for example [4]) that W corresponds to an optimality certificate, i.e., it satisfies

$$|B| = \rho^*_-(\cup_{i=1}^{r-1} V_i) = |(\cup_{i=1}^{r-1} V_i) \setminus W| + \sum_{i=1}^{r-1} \rho_{\Lambda_i}(W), \quad (7)$$

where ρ^*_- is the rank function of the matroid M^*_- and ρ_{Λ_i} is the rank function of M_{Λ_i} for $i \in [r-1]$. Let ϕ^* be the maximum flow value of the network. By Theorem 7, we have

$$\rho^*_-(\cup_{i=1}^{r-1} V_i) = |\cup_{i=2}^{r-1} V_i| + \phi^*. \quad (8)$$

Let $W_i = W \cap V_i$ for $i \in [r]$. Notice that $W_1 = V_1$ and $W_r = \emptyset$. By Proposition 3 we have

$$\sum_{i=1}^{r-1} \rho_{\Lambda_i}(W) = \sum_{i=1}^{r-1} (\lambda_i(W_i, V_{i+1} \setminus W_{i+1}) + |W_{i+1}|)$$
$$= \phi(W) + \sum_{i=2}^{r-1} |W_i|.$$

Combining this result with (7) and (8), we obtain

$$\phi^* = |V_1 \setminus W_1| + \phi(W) = \phi(W),$$

which implies that W is a minimum cut of the linking network. ■

V. Conclusion

We have introduced a new type of flow network, called linking network, and shown that it can be used to analyze the ADT model. In particular, using results on matroids and linking systems, properties like the max-flow min-cut duality easily follow. The problem of finding a maximum flow and a minimum cut was reduced to the matroid intersection or partition problem, which allows us to profit from well-established matroid optimization algorithms, leading to a faster algorithm for larger capacity networks. Another advantage of the presented approach using linking networks, is its generality, which allows to adapt or extend the ADT model. In particular, using a weighted matroid intersection algorithm we can efficiently find minimum cost flows in the ADT model when costs are introduced for using inputs and outputs.

References

[1] A. Amaudruz and C. Fragouli. Combinatorial algorithms for wireless information flow. In *SODA '09: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2009.

[2] A. S. Avestimehr, S. N. Diggavi, and D. N. C. Tse. A deterministic approach to wireless relay networks. In *Proceedings of Allerton Conference on Communication, Control, and Computing*, September 2007. http://licos.epfl.ch/index.php?p=research_projWNC.

[3] A. S. Avestimehr, S. N. Diggavi, and D. N. C. Tse. Wireless network information flow. In *Proceedings of Allerton Conference on Communication, Control, and Computing*, September 2007. http://licos.epfl.ch/index.php?p=research_projWNC.

[4] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley, New York, 1998.

[5] W.H. Cunningham. Improved bounds for matroid partition and intersection. *SIAM Journal on Computing*, 15:948–957, 1986.

[6] A. Frank. A weighted matroid intersection algorithm. *Journal of Algorithms*, 2:328–336, 1981.

[7] S. Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM Journal on Computing*, 32(4):833–840, 2003.

[8] S. Iwata and J. B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *SODA '09: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1230–1237, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[9] S. M. Sadegh Tabatabaei Yazdi and S. A. Savari. A combinatorial study of linear deterministic relay networks. http://arxiv.org/abs/0904.2401v1, 2009.

[10] A. Schrijver. *Matroids and Linking Systems*. PhD thesis, Mathematisch Centrum, 1978.

[11] A. Schrijver. Matroids and linking systems. *Journal on Combinatorial Theory, Series B*, 26(3):349–369, June 1979.

[12] A. Schrijver. *Combinatorial Optimization — Polyhedra and Efficiency*. Springer, 2003.

[13] M. Shigeno and S. Iwata. A dual approximation approach to weighted matroid intersection. *Operations Research Letters*, 18:153–156, 1995.