A Unification of Edmonds' Routing Theorem and Ahlswede et al's Network Coding Theorem

Yunnan Wu, *Student Member, IEEE,* Kamal Jain, Sun-Yuan Kung, *Fellow, IEEE.*

***Abstract*— Given a network of lossless links with rate constraints, a source node, and a set of destination nodes, the *multicast capacity* is the maximum rate that the source can transmit common information to the destinations. The multicast capacity cannot exceed the capacity of any cut separating the source from a destination. The minimum of the cut capacities is called the cut bound. A fundamental graph theory by Edmonds established that if all nodes other than the source are destinations, the cut bound can be achieved by routing. In general, however, the cut bound cannot be achieved by routing. Ahlswede et al. established that the cut bound can be achieved by performing network coding, which generalizes routing by allowing information to be mixed. This paper presents a unifying theory, which has Edmonds' Routing Theorem and Ahlswede et al.'s Network Coding Theorem as special cases. Specifically, it shows that the multicast capacity can still be achieved even though information mixing is only allowed on edges entering relay nodes.**

***Index Terms*— Network coding, routing, multicast, flow, capacity.**

I. Introduction

We consider the problem of information multicasting, namely transmitting common information from a source node s to a set of destination nodes T, in a network of lossless links with rate constraints (link capacities). The network can be represented by a directed graph $G = (V, E)$, where the vertex set V and the edge set E denote the nodes and links, respectively. It is assumed that each edge, say from v to w, can be used to transfer information from its *tail* v to its *head* w at unit-rate; multiple edges with the same tail and head are generally allowed, so as to represent different link capacities. Given G, s, and T, the *multicast capacity* refers to the maximum multicast rate from s to T.

An upper bound of the multicast capacity can be established by examining the *cuts* that separate s from any destination $t \in T$. For $t \in T$, a partition of the nodes $V = V_L + V_R$ with $s \in V_L$, $t \in V_R$ specifies an s-t-cut, which refers to the set of edges going from V_L to V_R. The *capacity* of the cut refers to the sum capacity of the constituent edges, which is equal to the cardinality of the cut since each edge has unit capacity. An s-t-cut bears the meaning of a bottleneck that limits the rate at which information can be transmitted from s to t. Let $C_G(s,t)$ denote the minimum capacity of an s-t-cut, and introduce

$$C_G(s,T) \equiv \min_{t \in T} C_G(s,t). \tag{1}$$

Then $C_G(s,T)$ is an upper-bound of the multicast capacity; it will be referred to as the *cut bound* in the sequel. Naturally, the question to ask is whether it is achievable by *routing*, i.e., having nodes in the network store-and-forward information.

In a very special case, where there is only a single destination node, i.e., $T = \{t\}$, a fundamental theorem in graph theory by Menger [3] shows that the cut bound $C_G(s,t)$ is achievable by routing.

> **Menger's Theorem on Packing Edge-Disjoint Paths (1927) [3]:** *In a directed graph G, the maximum number of edge-disjoint s-t-paths (i.e., paths from s to t) is $C_G(s,t)$.*

Specifically, the *unicast capacity* can be achieved by routing information along $C_G(s,t)$ edge-disjoint s-t-paths.

In another extreme case, where all nodes other than the source s are destinations, i.e., $T = V - \{s\}$, another fundamental theorem in graph theory, by Edmonds [4], shows that the cut bound $C_G(s,T)$ is achievable by routing.

> **Edmonds' Theorem on Packing Spanning Trees (1972) [4]:** *In a directed graph $G = (V,E)$, the maximum number of edge-disjoint spanning trees rooted at $s \in V$ is $C_G(s, V - \{s\})$.*

Specifically, the *broadcast capacity* can be achieved by routing information with $C_G(s, V - \{s\})$ edge-disjoint trees, each connecting s with all other nodes.

According to [5], in the graph theory literature, some conjectures were made regarding possible generalizations of Edmonds' Theorem to the general case, where there exist *Steiner nodes*; these conjectures have been disproved by examples however. (The Steiner nodes are nodes other than the source s and the destinations T. They are used as relays.) One conjecture is about the existence of $C_G(s,T)$ edge-disjoint *Steiner trees*, each connecting the source s with the destination nodes in T. However, a counter example (Fig. 1) was provided by Lovász [6] to show that there in general do not exist $C_G(s,T)$ edge-disjoint Steiner trees connecting s with T.

Recently, an example has been given in [7] to show that routing in general cannot achieve the multicast rate. This result is stronger than saying $C_G(s,T)$ edge-disjoint Steiner trees do

This work was presented in part at the Allerton Conference on Communications, Control, and Computing, Monticello, IL, Sept. 2004 [1], and the IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Philadelphia, PA, Mar. 2005 [2].

Yunnan Wu is with the Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544 (email: yunnanwu@princeton.edu).

Kamal Jain is with Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399 USA (email: kamalj@microsoft.com).

Sun-Yuan Kung is with the Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544 (email: kung@princeton.edu).

not generally exist.[1] In fact, Fig. 1 can also serve this purpose. Suppose a multicast rate of 2.0 can be achieved by routing. First, observe that e_1 and e_3 must carry different information. Next, note that in order for t_1 to have rate 2.0, the information carried by e_3 must be forwarded along the path $t_2 \rightarrow t_3 \rightarrow u \rightarrow t_1$. Thus the information on e_7 must be the same as the information on e_3. On the other hand, a symmetric argument with regard to t_2 shows that the information on e_7 must be the same as the information on e_1. This leads to a contradiction.

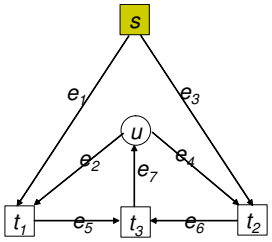

Fig. 1. An example graph G_1. The source node s is shown with a filled square, the destinations t_1, t_2, t_3 are shown with unfilled squares, and the only Steiner node u is shown with a circle. It can be checked that $C_{G_1}(s,t_1) = C_{G_1}(s,t_2) = C_{G_1}(s,t_3) = 2$ and $C_{G_1}(s,u) = 1$. Therefore, $C_{G_1}(s,T) = 2$ and $C_{G_1}(s, V - \{s\}) = 1$. This example was first given by Lovász [6] to show that there are no two edge-disjoint Steiner trees connecting s with T although $C_{G_1}(s,T) = 2$.

Note though, routing does not encompass all operations that can be performed at a node. Recently, the notion of *network coding* arises as a generalization of routing. Network coding refers to a scheme where a node is allowed to generate output symbols by encoding (i.e., computing certain functions of) the symbols it received. Thus, network coding allows information to be "mixed". This is in contrast to the routing pradigm, where each output symbol has to be a copy of one of the received symbols. In their pioneering work on network coding [7], Ahlswede *et al.* established that the cut bound $C_G(s,T)$ is achievable by network coding. Shortly afterwards, Li, Yeung, and Cai [9] showed that the multicast capacity can be achieved by linear network coding, i.e., using linear encoding functions at each node. Koetter and Médard [10] gave an algebraic characterization of linear encoding schemes and proved existence of linear time-invariant codes achieving the multicast capacity. The following statement summarizes these results.

> **Network Coding Theorems (2000-2003) [7], [9], [10]:** *In a directed graph $G = (V,E)$ with unit capacity edges, the multicast capacity from source s to destination nodes T is $C_G(s,T)$ and it can be achieved by performing (linear time-invariant) network coding.*

We now use the example graph G_1 to briefly explain the meaning of linear time-invariant (LTI) network coding. Assume the network operates as a synchronous system with a discrete time index n running from 0 to $+\infty$. The source node s generates two information symbols (e.g., two bits), x_{1n} and x_{2n}, in the n-th time unit. Let the z-transform ($D = z^{-1}$) of the two streams of symbols be $x_1(D) \equiv \sum_{n=0}^{+\infty} x_{1n} D^n$ and $x_2(D) \equiv \sum_{n=0}^{+\infty} x_{2n} D^n$. Assume it takes one time unit for a symbol to propagate through an edge. Let $G[s,h]$ be the graph obtained by adding to G a new vertex s' and h *source edges*, $s_1, \ldots, s_h$. Fig. 2(a) shows $G[s,2]$. Fig. 2(a) illustrates a LTI network coding solution achieving the multicast capacity. The information carried by each edge is shown beside each edge. The i-th source edge s_i carries the i-th stream of symbols $x_i(D)$; this represents how information is originally injected to the network. As shown in Fig. 2(a), $x_1(D)$ flows on e_1 and is then forwarded to e_5 after a unit delay, resulting in $x_1(D)D$ flowing on e_5; similarly, $x_2(D)D$ flows on e_6. With LTI network coding, the information flowing on an edge is generally allowed to be a causal linear combination of the information received on its incoming predecessor edges. For example, the received information from e_5 and e_6 is combined at t_3 to produce $[x_1(D)+x_2(D)]D^2$, which flows on e_7. Then, the mixed stream $[x_1(D)+x_2(D)]D^2$ is further forwarded to e_2 and e_4, after a unit delay. With this coding solution, destination t_1 receives $x_1(D)D$ from e_1 and $[x_1(D)+x_2(D)]D^4$ from e_2; it can then recover both source streams $x_1(D)$ and $x_2(D)$ after certain delay. Similarly, it can be checked that destination t_2 and t_3 can recover the source information. This confirms that this LTI coding solution achieves the multicast capacity.

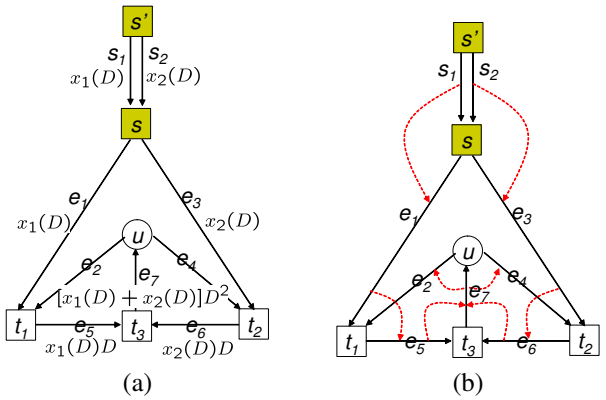

Fig. 2. (a) An example linear time-invariant network coding solution achieving the multicast capacity. The graph shown is $G[s,2] \equiv (V \cup \{s'\}, E \cup \{s_1, s_2\})$. (b) Coding relations among the edges, illustrated by dashed arrows.

Now let us take a more abstract look at how network coding is applied at each edge in this simple example. Fig. 2(b) shows the *coding relations* among the edges, determined by the LTI network coding solution of Fig. 2(a). We draw a dashed arrow from an edge f to e if $\mathrm{head}(f) = \mathrm{tail}(e)$ and f is involved in the linear combination producing the information flowing on e. Given a LTI coding solution, for each edge $e \in E$, the edges with dashed arrows pointing to e are said to be the *coding predecessors* of e. For example, the information flowing on e_5 is produced as a delayed version of the information flowing on e_1; thus e_5 has only one coding predecessor e_1. If an edge e has more than one coding predecessors, then the information flowing on e is produced by mixing information on the coding

[1] Finding the maximum rate for routing is equivalent to the problem of "fractionally packing Steiner trees" in graph theory; see, e.g., [8]. A fractional packing of Steiner trees refers to a weighted sum of Steiner trees where the weights can be fractional numbers. The distinction between fractional packing of Steiner trees and packing of edge-disjoint Steiner trees lies in that only binary weights are allowed in the latter.

predecessors.

It is seen from this example that mixing is in general required for optimality. Meanwhile, by Edmonds' Theorem, when all nodes other than the source node are destinations, no mixing is necessary to achieving the multicast capacity. Since network coding is a more complex operation than routing, Ahlswede et al's Theorem does not imply Edmonds' Theorem.

While network coding promises the highest rate, the use of mixing presumably incurs a higher processing complexity at the nodes, which is less desirable from a practical standpoint. For example, when $C_G(s, V-\{s\}) \approx C_G(s,T)$, there may be incentive to apply Edmonds' Theorem and construct a simpler routing-based solution that achieves rate $C_G(s, V-\{s\})$.

In this paper we show that it is possible to achieve the multicast capacity without paying the full complexity price, by using mixing only at some "critical places" and routing at others. Where are the critical places? In Fig. 2(b), information mixing is only performed at e_7. What distinguishes e_7 from other edges? In this example, only e_7 is entering a Steiner node and all other edges are entering destinations. We are thus motivated to classify the edges into two categories[2]: 1) edges entering Steiner nodes, which we call *Steiner edges*, 2) edges entering destination nodes, which we call *non-Steiner edges*. The main result of this paper is the following.

Theorem 1 (A Unifying Statement):
In a directed graph $G = (V, E)$ with unit capacity edges, the multicast capacity from source node s to destinations T, $C_G(s,T)$, is still achievable even though information mixing is only allowed on Steiner edges. More exactly, the multicast capacity can be achieved by performing (linear time-invariant) network coding subject to the constraint that each non-Steiner edge has at most one coding predecessor.

Theorem 1 implies Edmonds' Theorem and the network coding theorems as special cases. In the special case where all nodes other than the source are destinations, there is no Steiner edge and thus Theorem 1 says there is a capacity-achieving LTI network coding solution, where each edge has at most one coding predecessor. Then the information flowing on each edge e in the network is $y_e(D) = \alpha(D)x_i(D)$ for some $i \in \{1, \ldots, h\}$. We remove all edges with $y_e(D) = 0$ from $G[s,h]$. Think of a coloring of the edges in the resulting sub-graph of $G[s,h]$ with h color indices $\{1, \ldots, h\}$: if $y_e(D) = \alpha(D)x_i(D)$, then edge e is assigned the i-th color. In the capacity-achieving solution, each destination t has incoming edges spanning all h colors. By backtracking from the destinations, we can obtain h edge-disjoint spanning trees rooted at s. This establishes Edmonds' Theorem [3] as a special case.

As a generalization of the network coding theorems, the unifying statement has the following implications. On the theoretical side, this result advances the understanding of the very structural features of graphs that render network coding critical to achieving the multicast capacity. On the practical side, this result leads to savings in the processing complexity of a multicast system employing network coding such as in [11]–[14], without compromising the throughput.

[2]Without essential loss of generality, we assume source node s has no incoming edges. Thus the edge set can be partitioned into Steiner edges and non-Steiner edges.

Notations and Terminologies: For notational convenience, let $h \equiv C_G(s,T)$. Let $E_0 \equiv \{e : \text{head}(e) \in V(G[s,h]) - T\}$ and $E_1 \equiv \{e : \text{head}(e) \in T\}$ denote the set of Steiner edges and non-Steiner edges, respectively. Note that the source edges $s_1, \ldots, s_h$ are treated as Steiner edges as a notational convention. Having assumed source node s has no incoming edges in G, $E(G[s,h]) = E_0 + E_1$.

For a graph G, $V(G)$ and $E(G)$ denote its vertex set and edge set, respectively. For a directed graph $G = (V, E)$, let the edges going out of a vertex subset $X \subseteq V$ to its complement set $V - X$ be $\delta^{\text{out}}(X) \equiv \{e \in E : \text{tail}(e) \in X, \text{head}(e) \in V - X\}$. Let $\delta^{\text{in}}(X) \equiv \delta^{\text{out}}(V - X)$. We often do not distinguish one-element vertex sets from its single element, e.g., $\delta^{\text{out}}(v) \equiv \delta^{\text{out}}(\{v\})$. We use subscripts (e.g., $\delta^{\text{out}}_G(X)$) to specify the graph G when necessary.

II. An Algebraic Approach to Acyclic Graphs

The network coding theorems ensure the existence of a linear time-invariant network coding solution that achieves the multicast capacity. Theorem 1 says there exists a capacity-achieving network coding solution subject to additional coding constraints.

Thus, one natural thought is to start with an unconstrained capacity-achieving LTI coding solution and apply a series of local modifications to arrive at a solution that satisfies the additional constraint that edges entering destinations only use routing. In this section we first review linear network coding for acyclic graphs [9] and then discuss a simple algebraic argument that leads to a proof for acyclic graphs.[3]

For acyclic graphs, we can assume that edges have no delay without essential loss of generality and focus at one time slot, instead of discussing streams of data. Thus each edge $e \in E$ will carry one symbol, which we denote by y_e. The source edges $s_1, \ldots, s_h$ will carry the h source symbols $x_1, \ldots, x_h$, respectively.

In a linear network coding solution, the symbol on edge e is a linear combination of the symbols on the edges entering $\text{tail}(e)$, namely

$$y_e = \sum_{e':\ \text{head}(e')=\text{tail}(e)} w_{e,e'} y_{e'}, \tag{2}$$

where the coefficients $\{w_{e,e'}\}$ for a fixed edge e collectively specify the linear *local encoding operator* at e. By induction, y_e on any edge e is a linear combination of the source symbols, namely

$$y_e = \sum_{i=1}^{h} g_{e,i} x_i. \tag{3}$$

The vector $\mathbf{g}_e \equiv [g_{e,1}, \ldots, g_{e,h}]$ is known as the *global encoding vector* along edge e. It can be determined recursively

[3]As a historical note, the algebraic approach was established at the very beginning of this study, convincing us that Theorem 1 is likely to hold even for the cyclic case. However, it eventually turned out that a graph theoretic approach is more successful for the cyclic case. Nevertheless, it is our belief that the algebraic approach offers useful insights and has potential practical values in terms of distributed implementations.

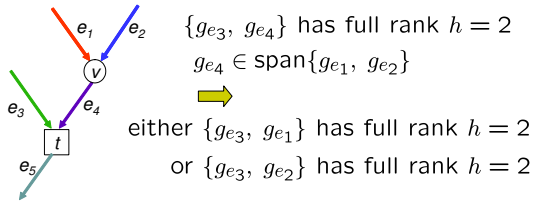

Fig. 3. An illustration of the algebraic approach.

as

$$\mathbf{g}_e = \sum_{e':\ \text{head}(e')=\text{tail}(e)} w_{e,e'}\mathbf{g}_{e'}, \tag{4}$$

where $\mathbf{g}_{s_i}$ is the ith unit vector $\boldsymbol{\epsilon}_i$. Since each y_e is a linear combination of the source symbols, any destination t receiving h symbols with linearly independent global encoding vectors can recover the source symbols.

A capacity-achieving linear network coding assignment is an assignment of local encoding operators $\{w_{e,e'}\}$ such that all destinations $t \in T$ can recover all the source symbols. A polynomial-complexity algorithm for finding a capacity-achieving linear network coding assignment has been given in [15], [16].

Note that a capacity-achieving linear network coding assignment for an acyclic graph $G = (V, E)$ exists if and only if there is a set of global encoding vectors $\{\mathbf{g}_e\}_{e\in E}$ that satisfy:

$$\mathbf{g}_e \in \text{span}\{\mathbf{g}_{e'},\ \text{head}(e') = \text{tail}(e)\}, \quad \forall e \in E; \tag{5}$$

$$\{\mathbf{g}_{e'},\ \text{head}(e') = t\} \text{ has full rank } h, \quad \forall t \in T. \tag{6}$$

Given an acyclic graph G, s, T, let $\{w_{e,e'}\}$ be a capacity-achieving linear network coding assignment. This determines the global encoding vectors $\{\mathbf{g}_e\}_{e\in E}$, which satisfy (5) and (6).

Visit the non-Steiner edges in E_1 one by one according to a topological order. When $e \in E_1$ is visited, try to enforce e to perform routing only by setting $\mathbf{g}_e := \mathbf{g}_{e'}$, for some predecessor edge e' with $\text{head}(e') = \text{tail}(e)$, while ensuring (5) and (6) are satisfied. We now explain this procedure using Fig. 3. Suppose the non-Steiner edge e_2 is currently being visited. From (6), $\{g_{e_3},\ g_{e_4}\}$ has full rank $h = 2$. From (5), $g_{e_4} \in \text{span}\{g_{e_1},\ g_{e_2}\}$. Therefore, either $\{g_{e_3},\ g_{e_1}\}$ has full rank, or $\{g_{e_3},\ g_{e_2}\}$ has full rank. Suppose $\{g_{e_3},\ g_{e_1}\}$ has full rank. Then we set $\mathbf{g}_{e_3} := \mathbf{g}_{e_1}$. After this change, condition (6) is still satisfied; condition (5) is satisfied for e_4. It remains to check that after the change, condition (5) is satisfied for all outgoing edges of $t = \text{head}(e_4)$, such as e_5; this holds since $\{g_{e_3},\ g_{e_1}\}$ has full rank.

III. A Graph Theoretic Approach to Cyclic Case

The algebraic argument mentioned above does not readily carry over to cyclic graphs. When the network has cycles, issues of delay and causality arise; in addition, the nodes do not have a topological ordering. With linear time invariant network coding, $\mathbf{g}_e$ becomes $\mathbf{g}_e(D)$ and the counterpart of (5) and (6) becomes more complicated. It is unclear whether it is possible to change the global encoding vectors one by one as done in the previous subsection.

We handle the complications due to cycles by resorting to graph theoretic approaches. We introduce a graph transformation called *hardwiring*. As illustrated in Fig. 6, hardwiring an edge e to edge f with $\text{head}(f) = \text{tail}(e)$ establishes a single predecessor for e; after this operation, the information on e has to come from f. We will then focus on establishing the following graph theoretic theorem.

Theorem 2 (Hardwiring Non-Steiner Edges): Given a directed graph G, a source node s, and destination nodes T, all the non-Steiner edges of G can be *hardwired* while preserving connectivity, i.e., in the resulting graph $\ddot{G}$, $C_{\ddot{G}}(s', T) = C_G(s, T)$.

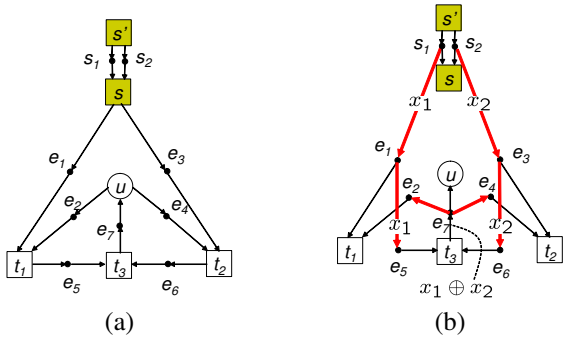

Fig. 4. (a) The "dotted" version $\dot{G}_1$ obtained by subdividing all edges of $G_1[s, 2]$. (b) A possible resulting graph $\ddot{G}_1$, after all non-Steiner edges have been hardwired.

For the example G_1 in Fig. 1, Fig. 4(b) gives a possible final graph $\ddot{G}_1$ satisfying the condition in Theorem 2.

In Section III-B, we will use Theorem 2 in conjunction with the network coding theorems to establish Theorem 1. We then establish Theorem 2 by proving the correctness of a procedure (Algorithm 1) that repeatedly hardwires a non-Steiner edge to a Steiner edge or another non-Steiner edge that has already been hardwired. The proof consists of two parts: 1) proving that each step is connectivity-preserving (Lemma 2), 2) proving that the procedure finishes hardwiring all the non-Steiner edges. The main challenge lies in the second part. For this part, we make use of a wire deletion operation that is more general than the hardwiring operation. Deleting a wire entering an edge e breaks one of its connections with the predecessor edges, whereas hardwiring e simultaneously breaks all but one connections with the predecessor edges. We prove by induction that all wires entering the non-Steiner edges that have not been hardwired can be deleted without losing the connectivity. The induction is performed in a special way: assuming that *any* subset of k wires across the cut can be deleted without losing the connectivity, we prove that any subset of $k+1$ wires across the cut can be deleted without losing the connectivity.

This procedure can be used as a constructive proof to Edmonds' Theorem, which is different from the existing approaches [4], [17], [22]–[26].

A. Some Graph Transformations

First we introduce for each edge of $G[s, h]$ a new node, shown as a dot in Fig. 4(a), which "divides" the edge into

two "halves". Denote the resulting graph by $\dot{G}$. Since there is a one-to-one mapping between a dot in $\dot{G}$ and an edge in $G[s,h]$, we refer to a dot in $V(\dot{G})$ (the vertex set of $\dot{G}$) by the name of the corresponding edge in $G[s,h]$. These dots play roles as the "connecting points" for the hardwiring operation.

As illustrated in Fig. 5, the operation $\texttt{expand}(e_1)$ introduces a new vertex v_{e_1} that has incoming edges from f_1, f_2, f_3 and an outgoing edge to e_1. Intuitively, this vertex v_{e_1} can be viewed as a duplicate of $v = \text{tail}(e_1)$ that provides information only for e_1. We use the name *wire* to refer to edges such as (f_1, v_{e_1}), (f_2, v_{e_1}), (f_3, v_{e_1}), since they correspond to connections between two edges in $G[s,h]$. Then, we introduce the operation $\texttt{deleteWire}$: after expanding e_1, $\texttt{deleteWire}(f_2, e_1)$ deletes (f_2, v_{e_1}), hence breaking the connection between f_2 and e_1.

Next we introduce a *hardwiring opeation*. As illustrated in Fig. 6, the operation $\texttt{hardwire}(f_1, e_1)$ deletes edge (v, e_1) and adds edge (f_1, e_1), where $v \equiv \text{tail}(e_1) \in V(G[s,h])$. Note that $\texttt{hardwire}(f_1, e_1)$ in Fig. 6 is equivalent to deleting all wires except (f_1, v_{e_1}) in Fig. 5. This operation enforces that the information on e_1 will only come from a single predecessor f_1, whereas previously the information on e_1 may be a mixture of the information on f_1, f_2, f_3. We use the name *hardwire* to denote an edge (f_1, e_1) after hardwiring e_1 with f_1, since earlier the potential predecessor edges of edge e_1 are f_1, f_2, f_3, whereas now the only predecessor edge of e_1 is f_1. As a degenerated case of the hardwiring operation, the operation $\texttt{hardwire}(\texttt{NULL}, e_1)$ simply deletes (v, e_1). After the operation $\texttt{hardwire}(f_1, e_1)$ (or $\texttt{hardwire}(\texttt{NULL}, e_1)$), we say the hardwired predecessor of e_1 is f_1 (or NULL) and write $\texttt{wiredPred}(e_1) = f_1$ (or $\texttt{wiredPred}(e_1) = \texttt{NULL}$).

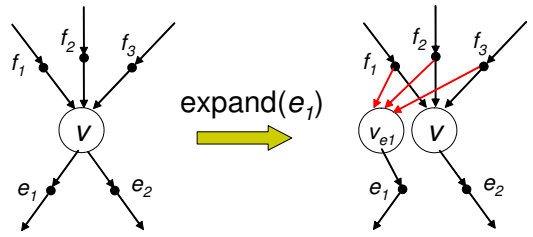

Fig. 5. The expanding operation.

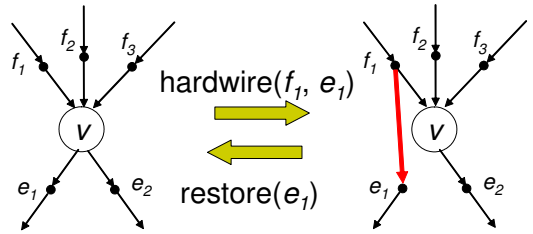

Fig. 6. The hardwiring operation and restoring operation.

We also introduce a *restoring operation*, which reverts the hardwiring operation back.

B. Proof of Theorem 1

We now establish Theorem 1 using Theorem 2 and the network coding theorems. Applying the network coding theorems to $\ddot{G}$, we see that there exists a (linear time-invariant) network coding solution achieving a rate $C_G(s,T)$ for multicasting from s' to t.

Since in $\ddot{G}$, each dot corresponding to a non-Steiner edge $e \in E_1$ has been hardwired to one of its predecessors, a LTI network coding solution on $\ddot{G}$ maps directly into a LTI network coding solution on G, which satisfies the additional constraint that each non-Steiner edge has only one coding predecessor. This can be easily verified on the example in Fig. 4(b). In other words, the structure of $\ddot{G}$ rules out the possibility of performing mixing to produce the information flowing on the non-Steiner edges. This establishes that the multicast capacity $C_G(s,T)$ can be achieved by LTI network coding subject to the constraint that each non-Steiner edge has only one coding predecessor.

C. A Hardwiring Procedure

Algorithm 1 performs a sequence of hardwiring operations on the graph $\dot{G}$ (as illustrated in Fig. 4(a)) to eventually arrive at a final graph $\ddot{G}$ (as illustrated in Fig. 4(b)), in which all non-Steiner edges E_1 have been hardwired. We use the notation $\dot{G}$ to refer to the "current" version $\dot{G}$, as the algorithm proceeds. Let $E_R \subseteq E_1$ denote the set of non-Steiner edges that have not been hardwired in the current $\dot{G}$. The procedure always checks if there exists a non-Steiner edge $e \in E_R$ that can be hardwired to a predecessor edge $f \in E_0 + E_1 - E_R$, i.e., a Steiner edge or a non-Steiner edge that has been hardwired, while preserving the required connectivity h to $\text{head}(e)$. If so, hardwire e to f and remove e from E_R. The algorithm stops when such a qualifying pair (f,e) cannot be found.

Algorithm 1 A Direct Hardwiring Algorithm

1: $E_R := E_1$;
2: **while** $\exists (f,e), f \in E_0 + E_1 - E_R, e \in E_R, \text{head}(f) = \text{tail}(e)$ such that after $\texttt{hardwire}(f,e)$, there are still h edge-disjoint paths to $\text{head}(e)$. **do**
3: $\quad \texttt{hardwire}(f,e)$;
4: $\quad E_R := E_R - e$;
5: **end while**

D. Proof of Correctness

Before proceeding to the main proof, we show a lemma, which is an easy consequence of Menger's Theorem (Max-Flow = Min-Cut). Fig. 7 illustrates the conditions in Lemma 1.

Lemma 1 (Pivoting Lemma):
In a directed graph $G = (V,E)$, consider three vertices s, v, and v', where s is not adjacent to either v or v'. Suppose there are h edge-disjoint paths $\mathcal{P}_1, \ldots, \mathcal{P}_h$, from s to v and there are h edge-disjoint paths $\mathcal{P}'_1, \ldots, \mathcal{P}'_h$, where $\mathcal{P}'_1$ is from v to v' and $\mathcal{P}'_2, \ldots, \mathcal{P}'_h$ are from s to v'. Then there are h edge-disjoint paths from s to v' in G.

Proof: Consider any s-v'-cut $\delta^{\text{out}}(X)$ with $s \in X$ and $v' \in V - X$. If $v \in V - X$, then the existence of h edge-disjoint s-v-paths implies that $|\delta^{\text{out}}(X)| \geq h$. If $v \in X$, then the existence of h edge-disjoint paths $\mathcal{P}'_1, \ldots, \mathcal{P}'_h$ going from X to $V - X$

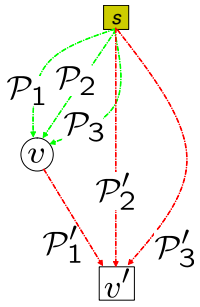

Fig. 7. The conditions in Lemma 1.

implies that $|\delta^{\text{out}}(X)| \geq h$. Thus any s-v'-cut contains at least h edges. By Menger's Theorem, there are h edge-disjoint s-v'-paths. ∎

We call the node v in Lemma 1 a *pivot* and the path $\mathcal{P}'_1$ a *partial path* since it is not from s to v'.

Lemma 2 (Connectivity preserving property): Consider a generic iteration of while-loop in Algorithm 1. Let $\dot{G}$ and $\dot{G}'$ denote the graphs before and after executing $\texttt{hardwire}(f, e)$, respectively. Assuming the existence of h edge-disjoint paths to each destination $t \in T$ in $\dot{G}$, then there exist h edge-disjoint paths to each destination $t \in T$ in $\dot{G}'$.

Proof: Due to the choice of wire (f, e), there are h edge-disjoint paths to $\text{head}(e)$ in $\dot{G}'$. For a destination $t \neq \text{head}(e)$, denote a set of h edge-disjoint paths to t in $\dot{G}$ by $\mathcal{P}_{t1}, \ldots, \mathcal{P}_{th}$. If e is not used in any of these paths, then $\mathcal{P}_{t1}, \ldots, \mathcal{P}_{th}$ are h edge-disjoint s'-t-paths in $\dot{G}'$. If e is used in one of these paths, say $\mathcal{P}_{t1}$, then $\mathcal{P}_{t1}$ has the following form

$$\mathcal{P}_{t1} = s \to \cdots \to e \to \text{head}(e) \to \cdots \to t. \tag{7}$$

This is because in Algorithm 1 an edge can only be hardwired to an edge in $E_0 + E_1 - E_R$. Thus when $e \in E_R$ is hardwired, there is no edge hardwired to e and hence e has only one outgoing edge to $\text{head}(e)$. Then we can apply the pivoting lemma to show $C_{\dot{G}'}(s', t) \geq h$, using $\text{head}(e)$ as the pivot, the sub-path of $\mathcal{P}_{t1}$ from $\text{head}(e)$ to t as the partial path $\mathcal{P}'_1$, and $\mathcal{P}_{t2}, \ldots, \mathcal{P}_{th}$ as $\mathcal{P}'_2, \ldots, \mathcal{P}'_h$. ∎

It remains to prove when Algorithm 1 stops, all the non-Steiner edges have been hardwired. We prove this by contradiction. Let E_R refer to the resulting set of non-Steiner edges that have not been hardwired. Suppose $E_R \neq \emptyset$.

We then *expand* all the edges in E_R. This allows us to apply the wire deletion operation, which has finer control granularity than the hardwiring operation. Denote the resulting graph by $\dot{G}$. Consider the cut

$$\delta^{\text{in}}(X) \equiv \{(u, v) \in E(\dot{G}) : u \in V(\dot{G}) - X, v \in X\}, \tag{8}$$

$$X = E_R \cup \{v_e : e \in E_R\}. \tag{9}$$

This is illustrated in Fig. 8, where all the edges that have not been hardwired reside in the right hand side.

We will prove that after deleting $\delta^{\text{in}}(X)$, there are still h edge-disjoint s'-t-paths, $\forall t \in T$. This will lead to a contradiction since it implies Algorithm 1 should have proceeded hardwiring these edges.

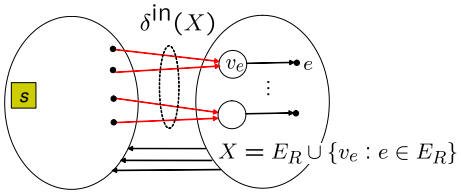

Fig. 8. A cut separating E_R from s.

We establish this result by inductively proving that deleting any subset of $\delta^{\text{in}}(X)$ consisting of k wires preserves the connectivity h. For $k = 0$, this is trivially true. Assuming this assertion is true for k, we now prove that after deleting an arbitrary subset $W \subseteq \delta^{\text{in}}(X)$ with $|W| = k+1$, there exist h edge-disjoint s'-t-paths, $\forall t \in T$. We prove the existence of h edge-disjoint paths first for destinations in

$$T_1 \equiv \{t \in T | \exists (f, v_e) \in W, \text{head}(e) = t\}, \tag{10}$$

and then for destinations in $T - T_1$, using the destinations in T_1 as pivots.

For any $t \in T_1$, choose a wire $(f, v_e) \in W$ with $\text{head}(e) = t$. By inductive assumption, there exist h edge-disjoint s'-t-paths $\mathcal{P}_{t1}, \ldots, \mathcal{P}_{th}$ in $\dot{G} - W + (f, v_e)$. If (f, v_e) is used in none of these paths, then $\mathcal{P}_{t1}, \ldots, \mathcal{P}_{th}$ are h edge-disjoint s'-t-paths in $\dot{G} - W$. If (f, v_e) is used in one of these paths, then it should have been possible to hardwire e to f in Algorithm 1. Thus a contradiction.

For any $t \in T - T_1$, pick an arbitrary wire $(f, v_{e_0}) \in W$. By inductive assumption, there exist h edge-disjoint s'-t-paths $\mathcal{P}_{t1}, \ldots, \mathcal{P}_{th}$ in $\dot{G} - W + (f, v_{e_0})$. If (f, v_{e_0}) is used in none of these paths, then $\mathcal{P}_{t1}, \ldots, \mathcal{P}_{th}$ are h edge-disjoint s'-t-paths in $\dot{G} - W$. If (f, v_{e_0}) is used in one of these paths, say $\mathcal{P}_{t1}$, then $\mathcal{P}_{t1}$ has the following form

$$\begin{aligned}\mathcal{P}_{t1} = s \to \cdots \to f \to \mathbf{v_{e_0}} \to \mathbf{e_0} \to \cdots \to \mathbf{v_{e_n}} \to \mathbf{e_n} \\ \to \text{head}(e_n) \to \cdots \to t.\end{aligned} \tag{11}$$

Note that the sub-path shown in boldface enters X via v_{e_0} and leaves X via e_n.

If $\text{head}(e_n) \in T_1$, then there exist h edge-disjoint paths to $\text{head}(e_n)$ in $\dot{G} - W$, according to our earlier discussions. Then we can apply the pivoting lemma to show $C_{\dot{G}-W}(s', t) \geq h$, using $\text{head}(e_n)$ as the pivot, the sub-path of $\mathcal{P}_{t1}$ from $\text{head}(e_n)$ to t as $\mathcal{P}'_1$, and $\mathcal{P}_{t2}, \ldots, \mathcal{P}_{th}$ as $\mathcal{P}'_2, \ldots, \mathcal{P}'_h$.

If $\text{head}(e_n) \notin T_1$, let m be the largest index in $\{0, \ldots, n\}$ such that $\text{head}(e_m) \in T_1$. This is well-defined since $(f, v_{e_0}) \in W$ and hence $\text{head}(e_0) \in T_1$. Since $\text{head}(e_{m+1}) \notin T_1$, none of the wires entering $v_{e_{m+1}}$ in $\dot{G}$ has been deleted in $\dot{G} - W$. Hence $v_{e_{m+1}}$ is essentially a duplicate of $\text{head}(e_m)$ in $\dot{G} - W$ (see Fig. 5). Then we can apply the pivoting lemma to show $C_{\dot{G}-W}(s', t) \geq h$, using $v_{e_{m+1}}$ as the pivot.

In other words,

$$\forall f : (f, \text{head}(e_m)) \in E(\dot{G} - W),\ (f, v_{e_{m+1}}) \in E(\dot{G} - W).$$

Hence in $\dot{G} - W$, the existence of h edge-disjoint paths to $\text{head}(e_m)$ implies the existence of h edge-disjoint paths to $v_{e_{m+1}}$.

E. Complexity

First, the number $h = C_G(s,T) = \min_{t\in T} C_G(s,t)$ can be found in $O(|T| \cdot h \cdot |E|)$, using Ford and Fulkerson's augmenting path method; see, e.g., [5]. For $i = 1, \ldots$, find the i-th augmenting path for all $t \in T$. Stop when the $h+1$-th augmenting path does not exist for some $t \in T$.

The running time of Algorithm 1 is dominated by the time required for the procedure calls of $\texttt{edgeDisjointPaths}(\dot{G}, s', t, h)$. Each such procedure call finds h edge-disjoint paths from s' to t in $\dot{G}$. Since $\dot{G}$ has at most $2|E|$ edges, $\texttt{edgeDisjointPaths}(\dot{G}, s', t, h)$ can be done in $O(h \cdot |E|)$ with an augmenting path method for finding maximum flow.

Algorithm 1 can have different implementations, depending on how the candidate wires (f, e) are enumerated and tested. For example, we can control the procedure to maximally grow one tree of hardwires before growing another one. Next, we present some observations that lead to low-complexity implementations.

First, note that we only need to test each pair (f, e) once, to check whether after $\texttt{hardwire}(f, e)$, there are h edge-disjoint paths to $\mathrm{head}(e)$. If the test result is negative, then testing $\texttt{hardwire}(f, e)$ later will give the same answer: e should not be hardwired to f. Further observe that for a given $e \in E_R$, testing the pairs

$$\{(f, e) | \, \mathrm{head}(f) = \mathrm{tail}(e), f \in E_0 + E_1 - E_R\} \qquad (12)$$

can be combined: in $\dot{G}$, disallow the connections between e and its predecessor edges in E_R and then try to find h edge-disjoint paths to $\mathrm{head}(e)$.

Based on these observations, we give a possible implementation of Algorithm 1 in Algorithm 2. In Algorithm 2, each edge in E_R has an associated boolean property, $\mathrm{active}(e)$, indicating whether the set (12) contains a pair that has not been tested. Initially, only the non-Steiner edges originated from s or a Steiner node are active. Then, we always consider the active edges in E_R to see if the set (12) contains a pair that can be hardwired while preserving connectivity. If the answer is yes, then e is hardwired and removed from E_R. In addition, this activates the successor edges of e in E_R, since the pair (e, e'), $e' \in E_R$, $\mathrm{head}(e) = \mathrm{tail}(e')$, has not been tested. If the answer is no, then e becomes inactive since the pairs in the current (12) have been tried.

Tong and Lawler [17] showed that a preprocessing step can be used to reduce the complexity for finding the maximum number of edge-disjoint spanning trees. They showed that given a directed graph $G = (V, E)$ and a root vertex s, a subgraph D that is the union of $C_G(s, V - s)$ edge-disjoint spanning trees rooted at s, can be found in time $O(C_G(s, V - s) \cdot |V| \cdot |E|)$. This preprocessing is as follows. For each non-root vertex v, find $C_G(s, V - s)$ edge-disjoint s-v-paths and delete the edges entering v that is on none of these paths. This preprocessing can also be applied to the current context to find a subgraph D of G with $C_D(s, T) = C_G(s, T)$ and the in-degree of each destination t is $h = C_G(s, T)$. For each destination $t \in T$, find h edge-disjoint s-t-paths and delete the edges entering t that is on none of these paths.

Algorithm 2 A Possible Implementation of Algorithm 1

```
E_R := E_1;
active(e) :=true, for all e : tail(e) ∈ V − T, head(e) ∈ T;
active(e) :=false, for all e : tail(e) ∈ T, head(e) ∈ T;
while ∃e : e ∈ E_R && active(e) :=true do
    expand(e);
    deleteWire(f', e), for all f' : f' ∈ E_R, head(f') = tail(e);
    [P_1, ..., P_h'] := edgeDisjointPaths(Ġ, s', head(e), h);
    restore(e);
    if h' == h then
        wiredPred(e) := predecessor(P_1 ∪ ... ∪ P_h, e);
        hardwire(wiredPred(e), e);
        E_R := E_R − e;
        active(e'):=true, for all e' : head(e) = tail(e'), e' ∈ E_R;
    else
        active(e):=false;
    end if
end while
```

With this preprocessing, the in-degree of each destination t is $h = C_G(s, T)$. Thus, for a given $e \in E_1$ with $\mathrm{tail}(e) \in T$, i.e., edges going from a destination node to another destination node, the number of pairs (f, e) to be tested is bounded by h

$$|\{(f, e) | \, \mathrm{head}(f) = \mathrm{tail}(e)\}| \le h. \qquad (13)$$

During the execution of Algorithm 2, these edges may switch between active and inactive state for at most h times. Thus, $\texttt{edgeDisjointPaths}(\dot{G}, s', \mathrm{head}(e), h)$ will be called at most h times. For a given $e \in E_1$ with $\mathrm{tail}(e) \in V - T$, i.e., edges going from a non-destination node to a destination node, all the pairs $\{(f, e) | \, \mathrm{head}(f) = \mathrm{tail}(e)\}$ will be tested in one call of $\texttt{edgeDisjointPaths}(\dot{G}, s', \mathrm{head}(e), h)$. Therefore, the total number of calls of $\texttt{edgeDisjointPaths}$ is $O(h \cdot |E_1|)$. Thus, the overall complexity is $O(h \cdot |E_1| \cdot h \cdot |E|)$.

Let G_T denote the sub-graph of G induced by the destinations T, i.e., $E(G_T) = \{e \in G | \, \mathrm{head}(e) \in T, \mathrm{tail}(e) \in T\}$. If G_T is acyclic, we can visit the non-Steiner edges by a topological order in Algorithm 2. This leads to a complexity of $O(\cdot|E_1| \cdot h \cdot |E|)$.

A different algorithm, with time complexity $O(|E_1| \cdot |T| \cdot h \cdot |E|)$ for general graphs, was presented in [1]. The algorithm of [1] visits the non-Steiner edges one by one in an arbitrary order and in each step, newly hardwires a non-Steiner edge and possibly re-hardwires some non-Steiner edges visited earlier.

IV. Discussions

A. Unification of Menger's Theorem, Edmonds' Theorem, and Ahlswede et al's Theorem

Corollary 1:
In a directed graph $G = (V, E)$ with unit capacity edges, the multicast capacity from sender s to receivers T is $C_G(s, T)$ and it can be achieved with (linear time-invariant) network coding subject to the following constraints:

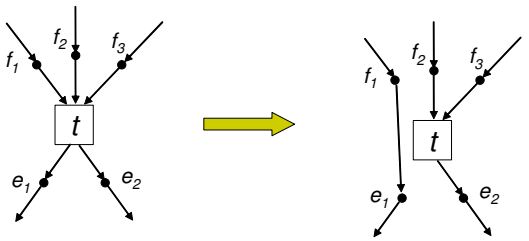

Fig. 9. The splitting off operation

- Each non-Steiner edge has at most one coding predecessor.
- Each edge has at most $|T|$ coding predecessors.

Proof: Theorem 1 says that there exists a LTI network coding solution subject to the first constraint. Let $\ddot{G}$ be the resulting graph after hardwiring all the non-Steiner edges of G, as in Theorem 2. To establish that a LTI network coding solution exists subject to both constraints, we need some extra steps that transforms $\ddot{G}$ into a graph G''. We then show that G'' has a network coding solution without coding constraints, which maps into a network coding solution of G satisfying the coding constraints.

Denote by G' the resulting graph after expanding all the Steiner edges. In G', there are still $C_G(s,T)$ edge-disjoint s'-t-paths, $\forall t \in T$. Then, we delete from G' all the wires that have not been used in these paths; denote the resulting graph by G''. According to the network coding theorems, there exists a (linear time-invariant) network coding solution that achieves a rate $C_G(s,T)$ for multicasting from s' to t in G''. This solution maps into a network coding solution for G, satisfying the coding constraints. ■

Corollary 1 unifies Menger's Theorem, Edmonds' Theorem, and the network coding theorems. When $T = \{t\}$, Corollary 1 states that there is a capacity-achieving LTI network coding solution, where each edge has at most one coding predecessor according to the second constraint. Similar to the discussions immediately following Theorem 1, by backtracking from destination t, we can find h edge-disjoint s-t-paths. This establishes Menger's Theorem as a special case.

B. Splitting Off Operation

Interestingly, the hardwiring operation introduced in this paper can be viewed as an asymmetric relaxation of a well-known *splitting off* operation in graph theory. In a directed graph, *splitting off* a pair of edges e_1, f_1 with $\mathrm{head}(f_1) = \mathrm{tail}(e_1)$ means that we replace e_1 and f_1 by a new edge $\bar{e}$ with $\mathrm{tail}(\bar{e}) = \mathrm{tail}(e), \mathrm{head}(\bar{e}) = \mathrm{head}(f)$. Fig. 9 illustrates this operation on $\dot{G}$.

The splitting-off operation was introduced by Lovász for undirected graphs [18]. Later Mader [19] derived some powerful results concerning splitting while preserving the local edge-connectivity in undirected graphs. Mader [20] also proved a splitting result for directed graphs. Under certain conditions, the splitting off operation is connectivity-preserving. The splitting off operation proves to be a very useful tool for establishing results about graph connectivity, especially for undirected graphs. For example, it has been used to derive a 2-factor approximation algorithm for packing Steiner trees in undirected graphs [8] and a generalized version of Edmonds' Theorem [21]. However, in the presence of Steiner nodes, it seems unclear whether the splitting off operation can be applied to prove Theorem 2, since its prior applications often require some global connectivity conditions to be fulfilled.

In comparison, the splitting off operation does not allow one-to-many local connection between an edge and its successors, whereas the hardwiring operation does. We look forward to future uses of this hardwiring operation to establish stronger results than those by the splitting off operation.

V. Conclusion

In this paper, we presented a statement that unifies and generalizes Edmonds' Theorem on routing and Ahlswede et al's Theorem on network coding. We classify the links in a network into two categories: 1) links entering relay nodes (Steiner edges), and 2) links entering destinations *non-Steiner edges*. We show the multicast capacity can be achieved by performing non-trivial network coding (mixing) only on the links entering relay nodes. In other words, links entering destinations will only require routing, which leads to a saving in the processing/implementation complexity.

We introduced a *hardwiring* operation: Hardwiring an edge e means restricting e to connect with at most one of its predecessor edges. We establish the unifying statement by proving that all non-Steiner edges can be hardwired while preserving the required connectivity to each destination $t \in T$. Our proof is accompanied by a procedure (Algorithm 1) that repeatedly hardwires a non-Steiner edge to a Steiner edge or another non-Steiner edge that has already been hardwired. This algorithm runs in time complexity $O(h \cdot |E_1| \cdot h \cdot |E|)$ for general graphs, and $O(|E_1| \cdot h \cdot |E|)$ for graphs in which the subgraph induced by destinations T is acyclic.

Acknowledgement

The authors would like to thank Dr. Philip A. Chou, Dr. László Lovász, and Dr. Tracey Ho for the helpful discussions with them.

References

[1] Y. Wu, K. Jain, and S.-Y. Kung, "A unification of Edmonds' graph theorem and Ahlswede et al's network coding theorem," in *Proc. 42nd Allerton Conf. Communication, Control and Computing*, Monticello, IL, Oct. 2004, invited paper.

[2] Y. Wu and S.-Y. Kung, "Reduced complexity network coding for multicasting over ad hoc networks," in *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, Philadelphia, PA, Mar. 2005.

[3] K. Menger, "Zur allgemeinen kurventheorie," *Fund. Math.*, vol. 10, pp. 95–115, 1927.

[4] J. Edmonds, "Edge-disjoint branchings," in *Combinatorial Algorithms, ed. R. Rustin*, Academic Press, NY, 1973, pp. 91–96.

[5] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003, vol. B.

[6] L. Lovász, "Connectivity in digraphs," *Journal of Combinational Theory B*, vol. 15, pp. 174–177, 1973.

[7] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Information Theory*, vol. IT-46, no. 4, pp. 1204–1216, July 2000.

[8] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing steiner trees," in *Proc. 14th Symp. on Discrete Algorithms (SODA)*. ACM-SIAM, 2003.

[9] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Information Theory*, vol. IT-49, no. 2, pp. 371–381, Feb. 2003.
[10] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.
[11] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. 41st Allerton Conf. Communication, Control and Computing*, Monticello, IL, Oct. 2003.
[12] Y. Wu, P. A. Chou, and K. Jain, "Practical network coding," Microsoft Research," Technical Report, in preparation.
[13] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. Int'l Symp. Information Theory*. Yokohama, Japan: IEEE, June 2003.
[14] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. 41st Allerton Conf. Communication, Control and Computing*, Monticello, IL, Oct. 2003.
[15] S. Jaggi, P. A. Chou, and K. Jain, "Low complexity optimal algebraic multicast codes," in *Proc. Int'l Symp. Information Theory*. Yokohama, Japan: IEEE, June 2003.
[16] P. Sander, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Symposium on Parallel Algorithms and Architectures (SPAA)*. San Diego, CA: ACM, June 2003, pp. 286–294.
[17] P. Tong and E. L. Lawler, "A faster algorithm for finding edge-disjoint branchings," *Information Processing Letters*, vol. 17, no. 2, pp. 73–76, Aug. 1983.
[18] L. Lovász, "On some connectivity properties of eulerian graphs," *Acta Math. Hungar.*, vol. 28, pp. 129–138, 1976.
[19] W. Mader, "A reduction method of edge-connectivity in graphs," in *Ann. Discrete Math.*, vol. 3, 1978, pp. 145–164.
[20] ——, "Konstruktion aller n-fach kantenzusammenhängenden digraphen," *Eupropean J. Combin.*, vol. 3, pp. 63–67, 1982.
[21] J. Bang-Jensen, A. Frank, and B. Jackson, "Preserving and increasing local edge-connectivity in mixed graphs," *SIAM J. Discrete Math.*, vol. 8, no. 2, pp. 155–178, 1995.
[22] R. E. Tarjan, "A good algorithm for edge-disjoint branching," *Information Processing Letters*, vol. 3, no. 2, pp. 51–53, Nov. 1974.
[23] L. Lovász, "On two minimax theorems in graph theory," *Journal of Combinational Theory B*, vol. 21, pp. 96–103, 1976.
[24] A. Frank, "Kernel systems of directed graphs," *Acta Sci. Math.*, vol. 41, pp. 63–76, 1979.
[25] W. Mader, "On n-edge-connected digraphs," in *Ann. Discrete Math.*, vol. 17, 1983, pp. 439–441.
[26] H. Gabow, "A matroid approach to finding edge connectivity and packing arboresences," in *Proc. 23rd ACM Symp. Theory of Computing*, 1991, pp. 112–122.