Beyond the Butterfly — A Graph-Theoretic Characterization of the Feasibility of Network Coding with Two Simple Unicast Sessions

Chih-Chun Wang
Center for Wireless Systems and Applications (CWSA)
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906
chihw@purdue.edu

Ness B. Shroff
Center for Wireless Systems and Applications (CWSA)
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47906
shroff@ecn.purdue.edu

***Abstract*— The problem of network coding with two simple unicast sessions is considered for general directed acyclic graphs. An explicit graph-theoretic characterization is provided for the feasibility of whether two symbols at different sources can be simultaneously transmitted to the designated sinks via network coding. The existence of a routing scheme is equivalent to finding edge-disjoint paths. Similarly, in this paper it is proven that the existence of a network coding scheme is equivalent to finding paths with controlled edge overlaps, and the characterization includes the well-studied butterfly graph as a special case. Various generalizations and implications are discussed based on the constructive nature of the flow-based conditions. For example, it is shown that a linear network coding scheme using only six paths is as effective as any non-linear network coding scheme.**

I. Introduction

Maximizing information exchange over communication networks has been a major subject among both the information theory and the networking societies. Network coding, a new paradigm allowing information duplicating and mixing in the intermediate nodes [1], [2], has demonstrated significant throughput advantage over routing algorithms, which traditionally regard information as unsplittable commodities.

For transmission with multiple coexisting sessions originated from different sources, coding has to be performed both within and across different sessions, and the benefit of network coding is clearly demonstrated in the butterfly graph of [1], [3]. In comparison to the single session scenario, the capacity region of multi-session network coding [4] has been less understood, and only for very special graphs can we characterize the capacity region, such as directed cycles [5], degree-2 three-layer directed acyclic networks [6], and special bipartite undirected graphs [5].

Necessary conditions for the existence of network coding solutions, also regarded as the outer bound of the capacity region, have recently been proposed based on the generalized *edge cut* condition of the underlying graph and the associated information-theoretic arguments, including fundamental regions in the entropy space [7], entropy calculus [8], the network-sharing bound [6], the information dominance condition [5], and the edge-cut bounds [9]. The achievability results, i.e., the inner bound of the capacity region, is generally determined by linear programming in a similar fashion to that of solving fractional multi-commodity flow, including the butterfly-based construction [10] and the pollution-treatment with powerset-based flow division [11]. The capacity region of a special *conservative* requirement is studied in [12].

We consider the simplest multi-session network coding over general directed acyclic networks, for which two symbols are transmitted simultaneously to the two designated sinks respectively. It is well known that the existence of a routing scheme is equivalent to finding edge-disjoint paths. In this paper, we prove an analogous result that the existence of a network coding scheme is equivalent to finding paths with *controlled edge overlaps*, and the characterization includes the well-studied butterfly graph as a special case. Our results show that when only inter-session coding is considered, the complexity of finding a good coding solution lies in finding the good paths rather than finding good encoding functions. The constructive nature of flow conditions also prompts new practical schemes realizing the promised improvement of network coding, including the *add-up-&-reset* scheme considered herein. Various implications and generalizations to linear/non-linear network coding and transmission of vectors of symbols are discussed as well.

II. The Main Result

A. Settings

We consider *finite, directed, acyclic* graphs (DAGs) $G = (V, E)$. An edge $e = (u, v) \in E$ is an ordered pair of nodes, for which u and v are termed as the tail and the head of e respectively. A path P from node u to v is defined as an ordered set of edges $\{(u, w_1), (w_1, w_2), \cdots, (w_n, v)\} \subseteq E$ such that the head of the previous edge is the tail of the next edge. We sometimes use the notation $P_{u,v}$ to emphasize the terminal nodes u and v. For a collection of paths $\mathcal{P} = \{P_1, \cdots, P_k\}$ and a given edge $e \in E$, the edge-share number

1-4244-1429-6/07/$25.00 ©2007 IEEE

of e is defined as $\mathsf{es}_{\mathcal{P}}(e) = |\{P \in \mathcal{P} : e \in P\}|$, i.e., the number of paths that use edge e.

We consider the following network coding problem with two simple unicast sessions: Given a finite DAG $G = (V, E)$ and two source-sink pairs (s_1, t_1) and (s_2, t_2), whether two single symbols X_1 and X_2, emanating from s_1 and s_2 respectively, can be "transmitted" to t_1 and t_2 simultaneously within *a single time slot*. It is assumed that each edge is capable of carrying only one symbol per time slot, and there is no transmission delay.

Both X_1 and X_2 are drawn from a finite field $\mathsf{GF}(q)$ with sufficiently large q, see [14], [15] and the reference therein. Since the size of q is not of our primary interest, the readers may safely assume that X_1 and X_2 take integer values instead, provided a sufficiently large q is adopted.

Throughout the paper, we also assume that both P_{s_1,t_1} and P_{s_2,t_2} exist, which can be checked within polynomial time. Otherwise simultaneous transmission is simply impossible.

In this paper, we focus solely on achieving simultaneous unsplittable transmission of two symbols within a single time slot, and therefore, time-sharing, or equivalently fractional routing/coding, is beyond the scope of consideration. The time-sharing issue will be briefly addressed in Section V-C.

B. The Main Result

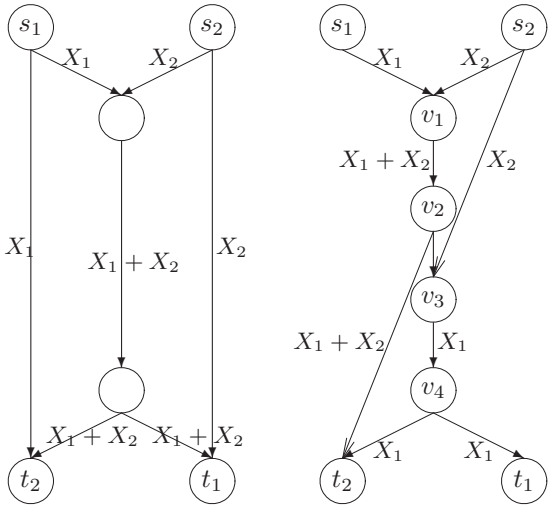
A graph-theoretic characterization on the feasibility of simultaneous transmission of two symbols in one time slot using *linear* network coding is provided as follows.

Theorem 1: A linear network coding scheme exists if and only if one of the following two conditions holds.

- [Condition 1] There exists a collection $\mathcal{P}$ of two paths P_{s_1,t_1} and P_{s_2,t_2}, such that $\max_{e\in E} \mathsf{es}_{\mathcal{P}}(e) \leq 1$.
- [Condition 2] There exist a collection $\mathcal{P}$ of three paths $\{P_{s_1,t_1}, P_{s_2,t_2}, P_{s_2,t_1}\}$, and a collection $\mathcal{Q}$ of three paths $\{Q_{s_1,t_1}, Q_{s_2,t_2}, Q_{s_1,t_2}\}$, such that $\max_{e\in E} \mathsf{es}_{\mathcal{P}}(e) \leq 2$ and $\max_{e\in E} \mathsf{es}_{\mathcal{Q}}(e) \leq 2$.

To our knowledge, *Theorem 1* is the first characterization of the feasibility of intersession linear network coding for general DAGs, which is based on the constructive flow conditions instead of the pessimistic cut conditions. *Theorem 1* serves as a first step toward generalizing the well understood min-cut max-flow characterization for the *multicast network coding* problem. An information-decomposition-based construction that simplifies the coding solution (similar to that for multicast network coding [16]) can thus be derived on the basis of this new necessary and sufficient condition.

The "if" and "only if" directions of *Theorem 1* will be discussed in the following sections respectively, in which we will use Conditions 1 and 2 to refer to the two separate conditions in *Theorem 1*. The generalization of *Theorem 1* to non-linear network coding will be provided in Section V-B.



(a) The butterfly. (b) The grail.

Fig. 1. Two examples of graphs benefited from network coding and their corresponding network coding schemes.

III. Sufficient Condition for Linear Network Coding

A. The Butterfly

If only traditional routing algorithms are allowed, the network transmission problem is equivalent to the 2-edge-disjoint-path problem characterized by Condition 1.

The benefit of adopting coding during the network transmission is clearly demonstrated by the simple butterfly graph (Fig. 1(a)) first depicted in the seminal paper of [1]. Most of the existing (theoretical or empirical) research developments rely on identifying the butterfly sub-structure within a graph to demonstrate the capacity gain [5], proving the inner bound of the capacity region [10], [6], or on devising practical schemes for realizing the promised throughput improvement [17].

Before proving the sufficiency of *Theorem 1*, we first provide a graph-theoretic condition (*Theorem 2*) in terms of paths and the corresponding edge-share number, such that any graph satisfying *Theorem 2* either admits routing algorithms or contains a butterfly as its substructure, the latter of which prompts the existence of good linear network coding schemes.

Theorem 2 (Butterfly Condition): A linear network coding scheme exists if one of the following two conditions holds.

- There exists a collection $\mathcal{P}$ of two paths P_{s_1,t_1} and P_{s_2,t_2}, such that $\max_{e\in E} \mathsf{es}_{\mathcal{P}}(e) \leq 1$.
- There exists a collection $\mathcal{P}$ of four paths P_{s_1,t_1}, P_{s_2,t_2}, P_{s_1,t_2}, and P_{s_2,t_1}, such that $\max_{e\in E} \mathsf{es}_{\mathcal{P}}(e) \leq 2$.

The second condition also guarantees that the underlying graph G contains a butterfly as its substructure.

Remark 1: By definition, *Theorem 2* can be viewed as a corollary of the strictly stronger *Theorem 1*. Hence the proof of *Theorem 2* is omitted.

Remark 2: Theorem 2 has converted the butterfly identification problem into a path finding problem similar to that for the traditional routing problem. Efficient coding schemes can

then be devised based on individual paths rather than on the butterfly substructure.

B. Beyond The Butterfly

As simple as the butterfly is, not all graphs suitable for network coding can be characterized by a butterfly or have a butterfly as its substructure. One simple example is the "grail graph" and its corresponding linear network coding scheme, as depicted in Fig. 1(b). It can be easily verified that the grail graph contains no butterfly as its substructure and it *does not* satisfy *Theorem 2*. On the other hand, the following choice of six paths satisfies Condition 2 in *Theorem 1*:

$$\begin{aligned} P_{s_1,t_1} &= s_1v_1v_2v_3v_4t_1 & Q_{s_1,t_1} &= s_1v_1v_2v_3v_4t_1 \\ P_{s_2,t_2} &= s_2v_1v_2t_2 & Q_{s_2,t_2} &= s_2v_3v_4t_2 \\ P_{s_2,t_1} &= s_2v_3v_4t_1 & Q_{s_1,t_2} &= s_1v_1v_2t_2, \end{aligned} \quad (1)$$

and *Theorem 1* guarantees the existence of a good network coding scheme, as also demonstrated in Fig. 1(b).

One key message of *Theorem 1* is that it is not necessary to find a "complete" butterfly to realize the throughput improvement of network coding. The existence of two "half" butterflies is sufficient and can be easily identified by finding six paths with controlled edge overlaps.

C. The Add-Up-&-Reset Scheme

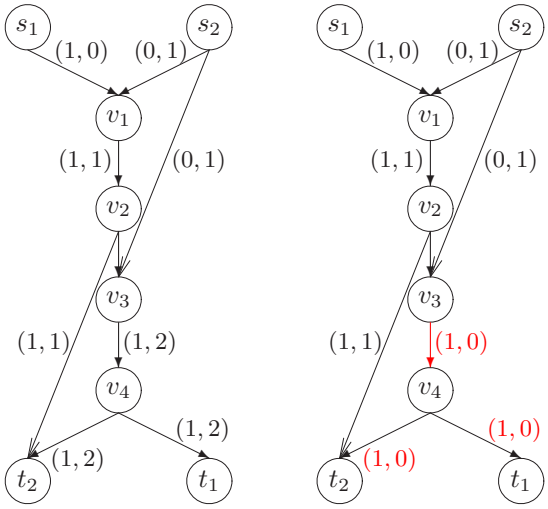
If Condition 1 in *Theorem 1* is satisfied, then a traditional routing scheme is sufficient for simultaneous transmission. Without loss of generality, we may thus assume that only Condition 2 is satisfied and prove the sufficiency of *Theorem 1* by constructing an add-up-&-reset scheme on the subgraph G' induced by the path collections $\mathcal{P}$ and $\mathcal{Q}$. All edges not involved in the six paths will remain inactive during the transmission. Serving only as a proof of the sufficiency, we do not *minimize* the path selections as did in [16].

As a linear network coding scheme, the message M along any edge is a linear combination $M = c_1X_1 + c_2X_2$ of the to-be-transmitted symbols X_1 and X_2, and we use the corresponding coding vector $M = (c_1, c_2)$ as shorthand. To ensure the computability of network coding, the outgoing message, as a two-dimensional vector, must be in the span of all incoming messages. We use M_e to denote the message along a specific edge e.

The "add-up stage" of the proposed add-up-&-reset scheme uses the following mixing rules for an edge e.

- If all the incoming messages of e are identical (including the case when there is only one incoming message), then m_e is identical to the incoming messages.
- Suppose one of the incoming messages (denoted by $M_1, \cdots, M_m$) is different. Choose M_e such that $M_e = a_1M_1 + \cdots + a_mM_m$ for some *strictly positive* integer coefficients a_1 to a_m, such that M_e is linearly independent of any other messages $M_{e'}$ for those e' not in the downstream of e.

Several properties of the add-up stage, as illustrated in Fig. 2(a), can be proved and are stated as follows.



(a) The add-up stage. (b) The complete add-up-&-reset scheme.

Fig. 2. The add-up-&-reset linear network coding scheme on the grail graph.

1) The integer coefficients described in the second rule always exist.
2) If two messages M_1 and M_2 are not identical, they are linearly independent.
3) For a message along a single edge e, its originating edge is defined as an upstream edge e' of e such that $M_e = M_{e'}$ and e' is the furthest away from e.[1] If two messages M_1 and M_2 are identical, they must have the same originating edge.
4) Along any edge that is reachable from s_i, the i-th component of the corresponding message must be a strictly positive integer.

A new "reset" operation is then introduced to complete the construction, for which we use Fig. 2(a) as an illustrative example. After the add-up stage, depending on whether each receiver is receiving different messages, we have the following three cases.

Case 1: Both t_1 and t_2 are receiving different messages. By the second property of the add-up stage, both t_1 and t_2 are able to *decode* X_1 and X_2 respectively, and the simultaneous transmission is achieved. The add-up stage alone is sufficient.

Case 2: Suppose only t_2 is receiving different messages generated from the add-up stage while the messages entering t_1 are identical, which is the case of Fig. 2(a). The latter condition implies the messages along the last edges of the three paths P_{s_1,t_1}, P_{s_2,t_1}, and Q_{s_1,t_1} must be identical. From the third property of the add-up stage, the three identical messages are originated from a common edge. We use (u, w) to denote the originating edge, which is (v_3, v_4) in Fig. 2(a). Then we perform the "reset-to-X_1" operation on (u, w), namely, the message along (u, w) is now reset to $(1, 0)$ as illustrated in Fig. 2(b). Properties 2 and 4 ensure that this reset operation

[1]The "originating edge" is similar to the root of the coding subtrees in the line graph discussed in [16].

is always possible. The messages along the downstream edges of (u, w) are then re-encoded according to the two rules of the add-up stage.

Since it is (u, w) that originates the three identical messages received by t_1, it can be easily shown that after applying the reset operation, t_1 receives X_1 perfectly, as illustrated in Fig. 2(b). However, the effect of the reset operation may ripple through the network and affect the received messages of t_2. We need some new properties of the add-up-&-reset scheme to complete the proof:

1) Property 1 of the add-up stage still holds.
2) Property 2 of the add-up stage still holds.
3) If two messages M_1 and M_2 are identical, they must have the same originating edges, or they must be of value $(1, 0)$.
4) Along any edge that is reachable from s_2 without using edge (u, w), the second component of the corresponding message must be a strictly positive integer.

If after the reset operation, t_2 still receives different messages, then by Property 2, t_2 is able to decode X_2.

If after the reset operation, t_2 begins to receive identical messages, which means either all its messages are originated from a common edge or all its messages have value $(1, 0)$ according to the new Property 3. The former case is not possible since the fact that t_2 was receiving different messages before applying the reset operation implies that the messages reaching t_2 must not have been originated from a single edge. The latter case is also impossible since Condition 2 in *Theorem 1* implies that P_{s_2,t_2} cannot use edge (u, w) and the new Property 4 thus guarantees a non-negative second component for the message along the last edge of P_{s_2,t_2}. From the above reasoning, t_2 must receive different messages and X_2 can be decoded.

Case 3: The last case is when both t_1 and t_2 receive identical messages during the add-up stage. Two reset operations will be performed. By analyzing the ripple effects of the two reset operations and the properties after the add-up and reset stages, it can be shown that the two reset operations will not interfere with each other and both sinks can decode the desired symbols successfully. The detailed proof is omitted due to the lack of space.

In practice, the add-up stage can be easily implemented by probabilistic coefficient assignments, and at most two reset operations will be performed. The simplicity of the add-up-&-reset scheme shows that similar to the routing problem, the bottleneck of the complexity of linear network coding resides in finding paths with minimal overlaps rather than in constructing effective encoding schemes.

IV. Necessary Condition for Linear Network Coding

Sketch of the proof of "⇒" in Theorem 1: We sketch the proof of the existence of $\mathcal{P}$ only and the existence of $\mathcal{Q}$ can be obtained by symmetry.

Since t_1 and t_2 are able to recover X_1 and X_2, the existence of P_{s_1,t_1} and P_{s_2,t_2} is certain. If P_{s_1,t_1} and P_{s_2,t_2} do not share any edge, then Condition 1 is satisfied. If they share any edge, then Condition 1 may not be satisfied but P_{s_2,t_1} must exist. Without loss of generality, the existence of P_{s_2,t_1} is assumed.

For the following, we consider a special path P_{s_2,t_2}, which is constructed by tracing the non-zero X_2 coefficient backward along the linear network coding. Take Fig. 1(b) as an example, the corresponding *backward* path is $P_{t_2,s_2} = t_2 v_2 v_1 s_2$.

Based on the above P_{s_2,t_2}, we will construct two paths P_{s_1,t_1} and P_{s_2,t_1} such that Condition 2 is satisfied. We first find arbitrarily a pair of paths $P^{(1)}_{s_1,t_1}$ and $P^{(1)}_{s_2,t_1}$. If Condition 2 is not satisfied, then there exists at least one violating edge e such that all three paths share e. If there exist more than one such e, these violating edges can be ordered by their positions in P_{s_2,t_2}. Pick the e that is the closest to t_2 and denote it by $e^{(1)}$. We can then construct another pair $P^{(2)}_{s_1,t_1}$ and $P^{(2)}_{s_2,t_1}$ based on $P^{(1)}_{s_1,t_1}$ and $P^{(1)}_{s_2,t_1}$ such that the closest violating edge $e^{(2)}$ (if exists) is strictly further away from t_2 than $e^{(1)}$. By repeatedly applying the above construction, we can find a pair $P^{(n)}_{s_1,t_1}$ and $P^{(n)}_{s_2,t_1}$ within a finite number of iterations such that Condition 2 is satisfied.

■

V. Implications & Generalizations

A. Complexity

Since the k-pair edge-disjoint path problem for DAGs can be solved in polynomial time [13], *Theorem 1* implies the following complexity result.

Corollary 1: For DAGs, deciding the feasibility of simultaneous transmission of two symbols via network coding can be solved in polynomial time.

Proof: The feasibility problem can be solved by replacing each edge by two parallel edges with the same head and tail and running the polynomial time 3-pair EDP algorithm [13] two times for $\mathcal{P}$ and $\mathcal{Q}$ respectively. ■

It is worth mentioning that the add-up-&-reset coding scheme can be decided in polynomial time as well. Decentralized coding scheme (given the path selections) is also possible by similar treatment of the information-decomposition technique in [16]. The NP-completeness for growing k, the number of source-sink pairs, is proved in [18].

B. Generalization to Non-Linear Network Coding

Theorem 1 can be further strengthened to non-linear network coding, provided that the special P_{s_2,t_2} used in the proof in Section IV is constructed via backward tracing messages with non-zero conditional mutual information $I(M; X_2|X_1)$. We then have the following theorem.

Theorem 3: Theorem 1 holds verbatim for non-linear network coding as well. Namely, a non-linear network coding scheme exists if and only if either Condition 1 or Condition 2 is satisfied.

The above theorem shows that in the simple setting considered herein, linear network coding is as effective as non-linear network coding.

Some corollaries of *Theorems 1* and *3* are provided as follows.

Corollary 2: If there exists a linear (or non-linear) coding scheme to transmit two symbols simultaneously in a finite DAG, there exists a linear network coding scheme using only six paths during transmission.

Corollary 3: The network-sharing bound stated in [*Theorem 1*, [6]] is tight in the simple setting considered herein. Namely, if the simultaneous transmission is not feasible, then there exists a cut for which the corresponding network sharing bound is violated.

C. Simple Generalization to Multiple Sessions

The 2-symbol simultaneous transmission problem can serve straightforward as a building block for more complicated cases described as follows. Consider n source-&-sink pairs and each source s_i would like to transmit d_i symbols to the corresponding sink t_i within one "time frame" over a finite DAG. Each edge is capable of transmitting c_e symbols per time frame with no transmission delay. Both d_i and c_e are positive integers. This setting includes the fractional routing/network coding (resulted by time-sharing) as special cases. One analogy of viewing time-sharing as a special coding scheme is that time-division multiple-access is simply a special case of code-division multiple-access.

We then have an inner bound of the capacity region, which is a strict improvement over the previous bound in [10].

Corollary 4: The rate vector $(d_1, \cdots, d_n)$ is feasible if the original graph G can be viewed as a superposition of one graph G_r and many graphs G_p's such that (i) routing is performed for every (s_i, t_i) pair in G_r, (ii) pairwise linear network coding across (s_i, t_i) and (s_j, t_j) is performed in each G_p individually, and (iii) the transmission rates $(d_1, \cdots, d_n)$ can be supported. The necessary and sufficient condition for G_r is the existence of a sufficiently large number of edge-disjoint paths, while the necessary and sufficient condition of each G_p is as stated in *Theorem 1*.

VI. Conclusion and Future Directions

In this paper, we have investigated the problem of network coding for the case of two simple unicast sessions for directed acyclic graphs, and provided an explicit characterization of the simultaneous transmissions of two symbols for different source and sink pairs. We have shown that the existence of a network coding scheme is equivalent to finding paths with controlled edge overlaps. Our results show that when only inter-session coding is considered, the complexity of network coding lies in finding good paths rather than finding good encoding functions. This result may pave the way for more practical solutions for implementing network coding. We now briefly outline some related topics that are outside the scope of this paper that we are currently investigating.

1) *Corollary 4* can certainly be improved by considering, in addition to pairwise inter-session coding, also the intro-session coding among symbols originated from the same source. In this case, we expect to have similar flow-based conditions, which can further push the inner bound toward the true capacity region.
2) The efficient computation of the inner bound in *Corollary 4* (analogous to that provided in [10]) is critical in terms of quantifying the benefit of network coding. A linear-programming based algorithm is a straightforward byproduct of our flow-based characterization *Theorem 1*.
3) An elegant practical "butterfly-seeking" implementation of network coding is provided in [17] to realize the capacity region within the inner bound of [10]. An immediate extension of this work will be to develop a path-based practical scheme that will fulfill the new achievable capacity inner bound of *Corollary 4*.
4) A major goal of our investigation is to develop efficient distributed linear network coding schemes for arbitrary numbers of source-sink unicast pairs. We believe that the approach described herein is a precursor to being able to characterize the achievable capacity for such systems.

References

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

[2] Z. Li and B. Li, "Network coding in undirected networks," in *Proc. 38th Conf. Inform. Sciences and Systems*. Princeton, NJ, USA, March 2004.

[3] S.-Y. R. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[4] Z. Li and B. Li, "Network coding: the case of multiple unicast sessions," in *Proc. 42nd Annual Allerton Conf. on Comm., Contr., and Computing*. Monticello, Illinois, USA, Sept. 2004.

[5] N. Harvey, R. Kleinberg, and A. Lehman, "On the capacity of information network," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2345–2364, June 2006.

[6] X. Yan, J. Yang, and Z. Zhang, "An outer bound for multisource multisink network codign with minimum cost consideration," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2373–2385, June 2006.

[7] L. Song, R. Yeung, and N. Cai, "Zero-error network coding for acyclic networks," *IEEE Trans. Inform. Theory*, vol. 49, no. 12, pp. 3129–3139, Dec. 2003.

[8] K. Jain, V. Vazirani, R. Yeung, and G. Yuval, "On the capacity of multiple unicast sessions in undirected graphs," in *Proc. Int'l Symp. Inform. Theory*. Seattle, USA, July 2006.

[9] G. Gramer and S. Savari, "Edge-cut bounds on network coding rates," *Journal of Network and Systems Management*, vol. 14, no. 1, pp. 49–67, March 2006.

[10] D. Traskov, N. Ratnakar, D. Lun, R. Koetter, and M. Médard, "Network coding for multiple unicasts: An approach based on linear optimization," in *Proc. Int'l Symp. Inform. Theory*. Seattle, USA, July 2006.

[11] Y. Wu, "On constructive multi-source network coding," in *Proc. Int'l Symp. Inform. Theory*. Seattle, USA, July 2006.

[12] N. Harvey, K. Jain, L. Lau, C. Nair, and Y. Wu, "Conservative network coding," in *Proc. 44th Annual Allerton Conf. on Comm., Contr., and Computing*. Monticello, IL, Sept. 2003.

[13] S. Fortune, J. Hopcroft, and J. Wyllie, "The directed subgraph homeomorphism problem," *Theoretical Computer Science*, vol. 10, pp. 111–121, 1980.

[14] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, October 2003.

[15] T. Ho, D. Karger, M. Médard, and R. Koetter, "Network coding from a network flow perspective," in *Proc. Int'l Symp. Inform. Theory*. Yokohama, Kanagawa, Japan, 2003.

[16] C. Fragouli and E. Soljanin, "Information flow decomposition for network coding," *IEEE Trans. Inform. Theory*, vol. 52, no. 3, pp. 829–848, March 2006.

[17] A. Eryilmaz and D. Lun, "Control for inter-session network coding," in *Proc. Workshop on Network Coding, Theory & Applications*, Jan. 2007.

[18] A. Lehman, "Network coding," Ph.D. dissertation, MIT, 2005.