Multi-terminal Function Multicasting in Undirected Graphs

Sreeram Kannan
Dept. of EECS,
UC, Berkeley
Email: ksreeram@berkeley.edu

Pramod Viswanath
CSL and Dept. of ECE
UIUC, IL 61801
Email: pramodv@illinois.edu

***Abstract*—In the function computation problem, certain nodes of an undirected graph have access to independent data, while certain other nodes of the graph require certain functions of the data; this model, motivated by sensor networks and cloud computing, is the focus of this paper. We study the maximum rates at which the function computation is possible on a capacitated graph.**

We consider a general model of function computation, which we term as multi-session function multicasting. In this general model, there are K independent sessions sharing the communication infrastructure; in each session, a set of D destinations all want the same function of a group of S sources. This traffic model generalizes various well known traffic models like the classical model of function computation($K = 1, D = 1$), multiple-unicasting ($S = 1, D = 1$) and multicasting ($K = 1, S = 1$).

For this general model, we propose a simple achievable strategy in which the function is computed for each session using Steiner trees at a specific destination and then distributed to the other destinations also using Steiner trees. Thus, in our proposed strategy, Steiner trees play the dual role of *computation trees* and also that of *multicasting trees*. Our main result is that this achievable strategy is *near optimal* for multi-session function multicasting of a wide class of functions in *undirected* graphs. The key technical contribution involves relating algorithmic work on Steiner cuts in undirected graphs to the function computation problem.

I. Introduction

In several communication scenarios, a receiver is interested in computing a function of data from different agents spread over a network. For example, in a sensor network, a fusion node is interested in computing a function of the various sensors. Similarly, in a cloud computing scenario, the data may be stored in a distributed manner, with different types of information about the same record stored in different locations. In such a scenario, one may be interested in computing a function of the data.

In this paper, we consider function computation in undirected graphs: there are K independent sessions; in each session D destinations all want the same function of S source variables. We consider the setting of block function computation, where receiver i is interested in computing R_iT function evaluations at the end of T time instants (the nodes in the network can forward arbitrary functions of data that they receive). The performance of the scheme is measured by the set of $(R_1, ..., R_K)$, which can be achieved, this is called the computation capacity region $\mathcal{C}$. This problem was originally formulated and studied in [11] where scaling laws for various functions was established in random graphs (further work along this lines was carried out in [22]). In this paper, we are interested in determining $\mathcal{C}$ for any specified graph.

We propose a simple achievable strategy for the problem of multi-session function multicasting. The strategy comprises of two phases: in the first phase, for each session, Steiner trees connecting the sources to a specific destination are used in order to perform computation of the function (so-called *computation trees*). In the second phase, Steiner trees connecting the various destinations of a session distribute this computed function to the other nodes. Thus this strategy separates *function computation* from *multicasting*, and also does not utilize inter-session coding, i.e., the information across distinct sessions is kept separate.

We consider functions of several variables, where each variable takes value in a fixed alphabet $\mathcal{A}$. A function is said to be λ-divisible if the function can be computed in a divide-and-conquer fashion with every intermediate computation requiring at most λ symbols over the alphabet to store and communicate. Alternately, every unit-capacity Steiner tree can be used to compute the function at rate $\frac{1}{\lambda}$. While every function on S variables has $\lambda \leq S$, we are interested in functions that have small λ. A function is said to be marginally injective, if it is injective with respective to every variable, given any realization of the remaining variables. Our main result is the following theorem, which characterizes the capacity region of this problem approximately.

Theorem 1. *For multi-session function multicasting of marginally-injective and λ_f- divisible functions, the proposed strategy achieves $\mathcal{R}_{ach}$ such that*

$$\frac{\bar{\mathcal{C}}}{2\lambda_f(S)g(S,K)} \subseteq \mathcal{R}_{ach} \subseteq \bar{\mathcal{C}}, \tag{1}$$

where $\bar{\mathcal{C}}$ is the rate region corresponding to a simple cut-set bound, and

$$g(S,K) = \begin{cases} 1 & \text{if } \mathcal{G} \text{ is a tree} \\ 2 & \text{if } K = 1, \\ \kappa(\log(S+D)K) & \text{if } K > 1, \end{cases} \tag{2}$$

where κ is a universal constant. Furthermore, the factor 2 in the left hand side of (1) can be dropped for the special case of either $S = 1$ or $D = 1$.

This scenario incorporates several special cases: when $S = 1, D = 1$, it is the multiple unicast problem with K-unicast, for which the seminal work of Leighton and Rao [12] shows a $O(\log K)$ gap between routing and cuts; when $K = 1, D = 1$, it is the function computation scenario with a single receiver being interested in a function of S nodes, when $K = 1, S = 1$, it is the multiple multicasting communication problem, where each source wants to multicast to a group of D receivers. We also consider as a special case here the case when the graph $\mathcal{G}$ is an undirected tree with $S \geq 1, D \geq 1$ and the sources and destinations located arbitrarily. Also, it is interesting to note that linear function computation, which has been well studied, is a special case with $\lambda_f = 1$. The duality between multicasting and linear function computation was observed in [18], however this is restricted to the case of linear coding. In this paper, we exploit the property that Steiner trees can be used either for multicasting or for function computation and furthermore show how to do the two things simultaneously.

A. *Prior Work*

A well-studied strategy in the context of single-session function computation, i.e., the case where $K = 1, D = 1$, is the use of *computation trees*, i.e., Steiner trees are packed and along the tree, computations are performed so that the function can be computed at the destination. This has been a very popular strategy, see [11] [6] [5] [23]. Such trees are optimal for function computation when $K = 1, D = 1$ and furthermore the graph is just a directed tree [6], [5]. For general directed graphs, the Steiner packing number can be far away from the cut. In undirected graphs, packing of specific computation trees, resembling the structure of the function, were considered and algorithms provided in [23]. However, no capacity results are available in this case.

Another popular achievable strategy for function computation is linear coding where each intermediate node forwards a linear combination of the incoming symbols. For general directed graphs with $D = 1, K = 1$, when the function to be computed is linear, random linear coding is known to be optimal [18]. However, if $D > 1$, even if the graph is undirected, it is easy to see that random linear coding performs poorly because different destinations will receive distinct linear combinations, whereas they demand the same linear combination. This pitfall is circumvented in our work by using structured coding schemes constructed based on Steiner trees.

The problem of linear function multicasting, i.e., the case of $S > 1, D > 1, K = 1$ with linear functions was studied in [19], [20], [21] for general directed graphs. In [21], several negative results demonstrating the insufficiency of scalar and vector linear coding is shown for directed graphs, deriving inspiration from similar results for multiple unicast in directed networks. In contrast, in this paper, it is shown that if the communication graph is undirected, simple coding schemes can achieve within a constant factor of the optimal even under the considerably general traffic model considered here.

The problem of determining function computation capacity in undirected graphs is considered harder in general, due to the presence of cycles in the graph, which allow for *interaction* in function computation. Single-shot function computation in the 2-node setting has been a central problem of study in the field of communication complexity [13], [14]. Even allowing for block computation, for general functions, seemingly simple 2-node problems can become hard (see [24]). For a class of sum-threshold functions on undirected trees, this capacity is characterized in [6] using carefully-orchestrated interactive strategies. In this paper, we show that for many function classes of practical interest, non-interactive function computation using computation trees can give near optimal performance.

II. Problem Formulation

The communication network is represented by an undirected graph $\mathcal{G} = (V, E)$, and edge capacity function $c : E \to \mathbb{R}^+$. There are K sources, session k has S sources:

$$\Sigma_k = \{\sigma_{1k}, \sigma_{2k} ..., \sigma_{Sk}\} \subseteq V, \quad k = 1, 2, ..., K, \tag{3}$$

and D destinations

$$P_k = \{\rho_{1k}, \rho_{2k}, ..., \rho_{Dk}\} \subseteq V \quad k = 1, 2, ..., K. \tag{4}$$

Define $G_k = \Sigma_k \cup P_k$ to be the group of terminals associated with session k. Each source σ_{ik} has several instances of an information variable X_{ik} over a common alphabet $\mathcal{A}$. For session k, each destination $\rho_{ik}, \quad \forall i$ wants to compute the *same* function of the sources $f(X_{1k}, ..., X_{Sk})$ taking values in alphabet $\mathcal{B}$. We do not impose any statistical structure on the messages and except the scheme to work under any possible assignment of the source variables. We assume that the source nodes do not overlap across sessions, i.e., each session is interested in a function of distinct source variables.

This function computation happens several times and therefore we consider a block function computation problem as follows. The network can potentially employ network coding (mixing of information inside or across sessions) in order to compute the function. Formally, a coding scheme of block length T and achieving a rate tuple $(R_1, ..., R_k)$ is described as follows:

- The source σ_{ik} has messages $X_{ik}(1), X_{ik}(2), ..., X_{ik}(R_k T)$. The destination ρ_k is interested in computing $f(X_{1k}(t), X_{2k}(t), ..., X_{Sk}(t))$ for $t = 1, 2, ..., R_k T$. The message set of source $v = \sigma_{ik}$ is denoted by W_{vk} where $|W_{vk}| = |\mathcal{A}|^{R_k T}$. For simplicity of notation, if a given node $v \notin \Sigma_k$, then we set $W_{vk} = \emptyset$.
- Since the network graph is undirected the capacity $c(e)$ on edge e has to be shared between the forward and reverse direction, let the fraction on forward direction be $\alpha(e)$. Once the α is fixed, then the network becomes a directed network. Let us scale this network to have integral capacities and represent this network as a *directed multigraph*, potentially having multiple edges between the same nodes.

TABLE I: Function Examples

	Function Name	$f(x_1,...,x_S)$	$\lambda_f(S)$-divisible	Marginally Injective?
1	Linear (field)	$x_1 \oplus x_2 \oplus ... x_S$	1	✓
2	Addition (abelian group)	$x_1 \oplus x_2 \oplus ... x_S$	1	✓
3	Arithmetic Sum ($A := \|\mathcal{A}\|$)	$x_1 + ... + x_S$	$\log_A\{S(A-1)+1\}$	✓
4	Real Sum over $\mathcal{A} = \{0:\delta:1\}$	$x_1 + ... + x_S$	$\log_A\{S(A-1)+1\}$	✓
5	ℓ_p-norm suitably quantized	$x_1^p + ... + x_S^p$	$\log_A\{S(A-1)+1\}$	✓
6	Histogram	Hist$(x_1,...,x_S)$	$\log_A\{\binom{S+A-1}{S}\}$	✓
7	Symmetric	f$(x_1,...,x_S)$	$\leq \log_A\{\binom{S+A-1}{S}\}$	Depends
8	Maximum	$\max(x_1,...,x_S)$	1	No

- At each node v, at each time t, there is a mapping
$$g_{v,t} : \mathcal{A}^{|\text{In}(v)|(t-1)} \times W_{v1} \times W_{vk} \times ... W_{vK} \to \mathcal{A}^{|\text{Out}(v)|},$$
where In(v) and Out(v) denote the set of incoming edges and the set of outgoing edges of node v. The function $g_{v,t}$ thus specifies a mapping from the messages on the incoming edges till time $t-1$ and the node's own messages to the outgoing edge message at time t.
- The decoding map at destination ρ_{ik} is given as a function of its incoming edges till time t
$$\psi : \mathcal{A}^{|\text{In}(v)|T} \to \mathcal{B}^{R_k T}. \quad (5)$$
- The coding scheme is said to achieve rate tuple $(R_1, ..., R_K)$ if each destination correctly recovers the function of its desired nodes.

The set of all achievable rate tuples $(R_1, ..., R_K)$ is called the computation capacity region $\mathcal{C}$.

A. Function Classes

We will define certain classes of functions which we will be interested in. A function $f : \mathcal{A}^S \to \mathcal{B}$ is called λ_f-divisible, if for every index set $I \subseteq [S]$, there exists a finite set $\mathcal{B}_I$ and a function $f^I : \mathcal{A}^{|I|} \to \mathcal{B}_I$ such that the following holds:

1) $f^{[S]} = f$
2) $|f^I(.)| \leq |\mathcal{A}|^{\lambda_f}$.
3) For every partition $\{I_1, ..., I_j\}$ of I, there exists a function $g : \mathcal{B}_{I_1} \times \cdot \times \mathcal{B}_{I_j} \to \mathcal{B}_I$ such that for every $x \in \mathcal{A}^{|I|}$,
$$f^I(x) = g(f^{I_1}(x_{I_1}), ..., f^{I_j}(x_{I_j})). \quad (6)$$

If a function is λ_f-divisible, then it means that it can be computed in a divide-and-conquer manner such that every intermediate computation needs only λ_f symbols to store and transmit. It must be noted that a function is λ_f-divisible implies that it can be divided in *any way* and still it will require at most λ_f symbols for storing intermediate computations. Every function is $|S|$-divisible in a trivial manner. Our interest will be in functions f for which λ_f is small. λ_f-divisible functions can be seen to be equivalent to λ_f-bounded functions defined in [5], we prefer the alternate name and definition since it is more suggestive. Divisible functions as defined in [11] [22] and [5] are λ_f-divisible with $\lambda_f = \log_{|\mathcal{A}|}|\mathcal{B}|$. For example, a linear function over a finite field has $\lambda_f = 1$. We refer the reader to Table I for a listing of λ_f for various functions (please see [1] for an elucidation of the table).

A function $f : \mathcal{A}^S \to \mathcal{B}$ is called *marginally injective* if, for any variable i, for any fixed assignment on the other variables, the function can take on $\mathcal{A}$ distinct values as x_i varies over $\mathcal{A}$, i.e.,
$$\forall i, \forall y_{[S]\setminus i} \in \mathcal{A}^{S-1}, \psi : \mathcal{A} \to \mathcal{B}, \quad (7)$$
defined by $\psi(x_i) = f(x_i, y_{[S]\setminus i}$, is injective.

Several functions of interest satisfy this property, as listed in Table I. An example of a function which is *not* marginally injective is the max-function over an ordered set. The value of the maximum does not depend on any other variable if one of the variable is assigned the maximum possible value.

III. Inner and Outer Bounds

Inner bounds based on computation trees and outer bounds based on cuts have been previously studied in the context of directed acyclic graphs with single destination node [5]. In this section, we present similar bounds in the context of multi-terminal function computation in undirected graphs.

A. Cut-set Bound

The cut-set bound defined in [5] is special to directed acyclic graphs and does not generalize to cyclic graphs. We use the technical condition of *marginally injectivity* in order to establish a simple cut bound on the communication rate. Define $G_k = \Sigma_k \cup P_k$. Given a set $\Omega \subseteq V$, define

$$K(\Omega) := \{k : G_k \cap \Omega \neq \emptyset, G_k \cap \Omega^c \neq \emptyset\}, \tag{8}$$

as the set of sessions disconnected by Ω and define

$$\text{Cut}(\Omega) := \sum_{(ij)\in E: i\in\Omega, j\in\Omega^c} c(ij). \tag{9}$$

Then, for any scheme computing a *marginally injective* function, for any $\Omega \subseteq V$, define

$$\bar{\mathcal{C}} = \{\overline{R} : \sum_{k\in K(\Omega)} R_k \leq \text{Cut}(\Omega)\ \forall\Omega\}. \tag{10}$$

The main observation is that $\mathcal{C} \subseteq \bar{\mathcal{C}}$. We will prove this for the case of $K = 1, D = 1$, the general case is similar. When $K = 1$, we only need to consider Ω such that $K(\Omega) = \{1\}$, i.e., the cut separates G_1. Let Ω be such that $\rho_{11} \in \Omega^c$ and for some i, $\sigma_{i1} \in \Omega$. The information on edges between Ω and Ω^c can take $\mathcal{A}^{\text{Cut}(\Omega)T}$ possible values. Since the function is marginally injective, it means that this should at least convey as much information as one of the sources, σ_{i1}. Thus $|\mathcal{A}|^{\text{Cut}(\Omega)T} \geq |\mathcal{A}|^{R_1 T}$, and so $R_1 \leq \text{Cut}(\Omega)$ which proves the required bound.

B. Achievable Strategy

We first formally define Steiner trees. An *undirected Steiner tree* on a set G is defined as an undirected tree τ which includes G in its vertex set. A *directed Steiner tree* rooted at ρ on a vertex set S is defined as a directed tree rooted at ρ that includes S in its vertex set. Given an undirected Steiner tree on G_k and an arbitrary node $g \in G_k$, there is a unique orientation of edges such that we get a directed Steiner tree rooted at g. Note that whenever we talk about Steiner tree, we talk about a tree with unit capacity edges.

To explain the proposed strategy, we first work with the case of single session, $K = 1$. The key idea of the proposed strategy is to break down the problem of function multicasting into two distinct tasks:

- First, the requisite function is computed at one of the destination nodes. This is done using a computation tree constructed from a Steiner packing.
- Second, the computed function is multicast to all the other destination nodes.

For doing the first task, we need a Steiner tree connecting the sources to a specific destination. For accomplishing the second task, we need a Steiner tree connecting all the destinations. In order to find the best achievable rate among all such strategies, we may have to further optimize the particular node at which the function is initially computed. This optimization problem may turn out to be hard to solve and therefore we resort to a further simplification.

In our proposed scheme, the destination at which the function is computed in the first stage is not optimized but chosen arbitrarily from P_k, let us fix it to ρ_{1k}, the first element of the set P_k. We first construct a Steiner tree between all the sources and all the destinations, i.e., a Steiner tree spanning the set G_k of nodes. We then use this Steiner tree in two phases to do function multicasting. In the first phase, the Steiner tree is used to compute the function at ρ_{1k}. To do this, we use the optimal strategy for function computation on this Steiner tree described in [5], [6], which basically forwards the sufficient statistic of the function based on the current variables in the tree. Thus, a Steiner tree of rate $\lambda_f(S)$ is sufficient to function computation at rate 1. Next, we use the same Steiner tree to do multicasting of the function. The function takes value in the alphabet $\mathcal{B}$ while the source symbols are from an alphabet $\mathcal{A}$. Thus, if we are given a Steiner tree of rate $\log_{|\mathcal{A}|}|\mathcal{B}|$, we can multicast the computed function at rate 1. Thus, in order to do function multicasting at rate 1, we need a Steiner tree of rate $\lambda_f(S) + \log_{|\mathcal{A}|}|\mathcal{B}| \leq 2\lambda_f(S)$. Stated differently, given a Steiner tree of rate 1, we can do function multicasting at rate $\frac{1}{\lambda_f(S)+\log_{|\mathcal{A}|}|\mathcal{B}|} \geq \frac{1}{2\lambda_f(S)}$.

In order to deal with the case of general K, we need to consider simultaneous Steiner packings for various subsets. Let $\mathcal{T}_k$ be the set of Steiner trees on G_k and $\mathcal{T} = \cup_k \mathcal{T}_k$. A fractional Steiner packing of $\pi_1, ..., \pi_K$ is said to be achievable if so many trees can be packed simultaneously: i.e., there exist $u_\tau, \tau \in \mathcal{T}$ such that

$$\pi_k = \sum_{\tau\in\mathcal{T}_k} u_\tau \quad k \in [K] \tag{11}$$

$$\sum_{\tau\in\mathcal{T}: e\in\tau} u_\tau \leq c(e) \quad \forall e \in E. \tag{12}$$

Thus given a fractional Steiner packing $(\pi_1, ..., \pi_K)$, the proposed scheme can achieve a rate tuple $\frac{1}{2\lambda_f(S)}(\pi_1, ..., \pi_K)$ for computing any λ_f*-divisible function*.

IV. Main Result

Sketch: In order to prove the main result (Theorem 1) for general K, we first give a description of rate regions using the max-concurrent flow representation, which is standard in the algorithmic literature. Then we compute the dual of this linear program. This dual is used to prove that there is no gap between Steiner tree packing and cuts in tree networks. Finally, we connect to existing results that imply logarithmic gaps between Steiner packings and cuts. Furthermore, there exist polynomial time algorithms, already in the literature, that achieve these rate guarantees.

We will now prove this result for $K = 1$ by connecting to the multicasting problem [7] and for general K using connections to algorithmic work showing approximation algorithms for "sparsest-Steiner-cuts" [2], [3], [4]. We refer the reader to the journal version [1] for a proof of the tree network result and for a detailed proof of the other results.

A. Single Session

In this setup, one node wants to compute a function of all the sources. Since $K = 1$, the regions collapse to single numbers, for which we use small case letters. There is a close relationship between the fractional Steiner packing number π and the cut (which is also called Steiner cut $\bar{c}$). This relationship is based on the Tutte-Nash-Williams theorem [8] [9] and Mader's undirected splitting-off theorem [10] and was elucidated in the multicast setting by Li and Li [7]:

$$\frac{1}{2}\bar{c} \leq \pi \leq \bar{c}. \tag{13}$$

Therefore, by using this fractional Steiner packing, and $r_{\text{ach}} = \frac{1}{\lambda_f}\pi$, we get the desired result,

$$\frac{1}{2\lambda_f}\bar{c} \leq r_{\text{ach}}. \tag{14}$$

B. General K

For the case of multiple sessions, we have to deal with rate regions and cut regions. In order to deal with these regions in a compact manner, we use the max-concurrent flow representation. Let $(D_1, ..., D_K)$ be a certain direction in the rate region, we can think of D_k as being demand for session k, where we want to achieve a rate pair $(R_1, ..., R_K) = \alpha(D_1, ..., D_K)$. In this representation, we would like to $\max \alpha$ such that $\alpha(D_1, ..., D_K) \in \mathcal{R}$. The achievable rate is given by $\frac{1}{2\lambda_f}(\pi_1, ..., \pi_K)$, where $(\pi_1, ..., \pi_K)$ is a simultaneous fractional Steiner packing. If we set $(\pi_1, ..., \pi_K) = \gamma(D_1, ..., D_K)$ and want to maximize γ, this problem is called the maximum concurrent Steiner flow problem, which can be written as $\max \gamma$ such that

$$\begin{aligned} \sum_{\tau \in \mathcal{T}_k} u_\tau &\geq \gamma D_k \quad \forall k \in [K] \\ \sum_{\tau \in \mathcal{T}: e \in \tau} u_\tau &\leq c(e) \quad \forall e \in E. \end{aligned}$$

The cut-set bound on $\mathcal{C}$ now translates to : $\alpha \leq \frac{\text{Cut}(\Omega)}{D(\Omega)}$. Thus $\alpha \leq \nu := \min_{\Omega \subseteq V} \frac{\text{Cut}(\Omega)}{D(\Omega)}$ is called the sparsest Steiner cut. Similarly, it is easy to see [2] that γ is also upper bounded by ν. There is an approximate max-Steiner-flow min-Steiner-cut theorem that relates the two quantities [2], [3], [4]:

$$\frac{1}{\kappa(\log SK)}\nu \leq \gamma \leq \nu, \tag{15}$$

where κ is a universal constant.

We can thus claim that, for computation of marginally-injective, λ_f-divisible functions that

$$\frac{1}{\lambda_f \kappa(\log SK)}\nu \leq \alpha \leq \nu. \tag{16}$$

Therefore we get

$$\frac{1}{\lambda_f \kappa(\log SK)}\bar{\mathcal{C}} \subseteq \mathcal{R}_{\text{ach}} \subseteq \mathcal{C} \subseteq \bar{\mathcal{C}}, \tag{17}$$

which proves the achievable portion of the result.

C. Algorithms

The results for capacity approximations are based on Steiner tree packings. However, it is well known that Steiner tree packing with even a single session is NP-hard in general undirected graphs. We would like to point out that, while the exact Steiner packing problem may be hard, there is a polynomial time algorithm (see Theorem 2.1 in [2]) that can provide a factor 2 approximation for the general multiple-session problem.

References

[1] S. Kannan and P. Viswanath, "Multi-session Function Computation and Multicasting in Undirected Graphs," *Jour. Sel. Areas in Commun.,* to appear.

[2] P. N. Klein, S. A. Plotkin, S. Rao, and E. Tardos, "Approximation Algorithms for Steiner and Directed Multicuts," *J. Algorithms* vol. 22, No.2, pp. 241-269, 1997.

[3] V. Nagarajan and R. Ravi, "Approximation Algorithms for requirement cut on graphs," *Algorithmica* vol. 56, no.2, pp.198-213, 2010.

[4] A. Gupta, V. Nagarajan and R. Ravi, "An improved approximation algorithms for requirement cut," *Operations Research Letters* vol. 38, pp.322-325, 2010.

[5] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: Cut-set bounds", *IEEE Trans. Information Theory*, vol. 57, no.2, pp. 1015 - 1030, Feb. 2011.

[6] H. Kowshik and P. R. Kumar, "Optimal Function Computation in Directed and Undirected Graphs," Available online: http://arxiv.org/abs/1105.0240

[7] Z. Li and B. Li. "Network Coding: The Case of Multiple Unicast Sessions," *Proc. 42nd Allerton Conference,* Monticello, IL, 2004.

[8] W. T. Tutte. "On the Problem of Decomposing a Graph Into n Connected Factors," *J. London Math. Soc.*, vol. 36, pp. 221230, 1961.

[9] C. St. J. A. Nash-Williams, "Edge-disjoint Spanning Trees of Finite Graphs," *J. London Math. Soc.,* vol. 36, pp. 445450, 1961.

[10] A. Frank, "Edge-connection of Graphs, Digraphs, and Hypergraphs," *Tech.Rep., Enervary Research Group on Combinatorial Optimization*, Budapest, Hungary, http://www.cs.elte.hu/egres, September 2000.

[11] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communication*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[12] F. T. Leighton and S. Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM* vol. 46, No.6, pp. 787-832, 1999.

[13] A. C. Yao, "Some complexity questions related to distributive computing," in *Proceedings of the eleventh annual ACM Symposium on Theory of Computing*, 1979, pp. 209–213.

[14] E. Kushilevitz and N. Nisan, *Communication Complexity*. Cambridge University Press, 1997.

[15] A. Orlitsky and J. R. Roche. Coding for computing. *IEEE Transactions on Information Theory*, 47:903–917, 2001.

[16] N. Ma, P. Ishwar, and P. Gupta, "Information-theoretic bounds for multiround function computation in collocated networks," in *Proceedings of the IEEE International Symposium on Information Theory*, 2009, pp. 2306–2310.

[17] V. Doshi, D. Shah, and M. Medard, "Source coding with distortion through graph coloring," *Proceedings of the IEEE International Symposium on Information Theory*, 2007, pp. 1501–1505.

[18] B. K. Rai, B. K. Dey, and S. Shenvi, "Some bounds on the capacity of communicating the sum of sources," in *ITW 2010, Cairo*, 2010.

[19] A. Ramamoorthy, "Communicating the sum of sources over a network," in *Proceedings of the IEEE International Symposium on Information Theory*, 2008, pp. 1646–1650.

[20] M. Langberg and A. Ramamoorthy, "Communicating the sum of sources in a 3-sources/3-terminals network," in *Proceedings of the IEEE International Symposium on Information Theory*, 2009, pp. 2121–2125.

[21] B. K. Rai, and B. K. Dey, "Sum-networks: System of polynomial equations, unachievability of coding capacity, reversibility, insufficiency of linear network coding," http://arxiv.org/abs/0906.0695, 2009.

[22] S. Subramanian, P. Gupta, and S. Shakkottai, "Scaling bounds for function computation over large networks," in *Proceedings of the IEEE International Symposium on Information Theory*, 2007, pp. 136–140.

[23] V. Shah, B. Dey, and D. Manjunath, "Network Flows for Functions, " *Proc. ISIT 2011*, St. Petersburg, Russia, Aug. 2011.

[24] R. Ahlswede and Ning Cai. On communication complexity of vector-valued functions. *IEEE Transactions on Information Theory*, 40:2062–2067, 1994.

[25] A. Anandkumar, M. Wang, L. Tong, and A. Swami. Prize-Collecting Data Fusion for Cost-Performance Tradeoff in Distributed Inference. *Proc. of IEEE INFOCOM,* Rio De Janeiro, Brazil, Apr. 2009.