Distributed Rate Allocation for Network-Coded Systems

Amin Jafarian, Sang Hyun Lee, and Sriram Vishwanath
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712, USA
Email: {jafarian,shlee,sriram}@ece.utexas.edu

Christina Fragouli
School of Computer and Communication Sciences
Ecole Polytechnique Federale de Lausanne (EPFL)
Lausanne, Switzerland
E-mail: christina.fragouli@epfl.ch

***Abstract*—This paper addresses the problem of distributed rate allocation for a class of multicast networks employing linear network coding. The goal is to minimize the cost (for example, the sum rates allocated to each link in the network) while satisfying a multicast rate requirement for each destination in the network. In essence, this paper aims to achieve network capacity while ensuring that the cost of operation (equivalently, the rate allocated per link in the network) is minimal.**

This paper uses a belief propagation framework to obtain a distributed algorithm for the rate allocation problem. Simulation results are presented to demonstrate the convergence of this algorithm to the optimal rate allocation solution.[1]

I. Introduction

It is well known that linear network coding achieves the capacity of a multicast network [3]. However, there may be multiple rate allocation strategies (rates assigned to each link in the network) that all achieve network capacity. An important problem is determining that rate allocation which achieves multicast capacity as "efficiently" as possible. By associating a cost with each link of the network (which is proportional to the rate allocated to each link), the problem of efficient rate allocation translates to determining the minimum cost rate allocation strategy for the network. Associating costs with network links is especially relevant in wireless communications, where it is important that energy and/or bandwidth be as efficiently utilized as possible.

The problem of network coding with cost, where a cost is associated with each link in the network has been extensively studied [6], [7]. It has been shown that the minimum cost solution can be formulated as a linear programming (LP), and for more general utility functions, a convex program. In general, this LP must be solved using a centralized solver. The goal of this paper is to derive a belief propagation (BP) based decentralized algorithm to solve this optimal rate allocation.

For networks employing routing instead of network coding, rate allocation is often formulated as an integer linear programming (ILP) [2], [5], [8]. In general, this ILP is computationally hard to solve. In recent years, both centralized and distributed (approximate) solvers for ILP's have been developed [1]. The distributed solver based on BP in [9] for ILP's forms the basis for our work in this paper.

[1]We thank the generous support we receive from the DARPA IAMANET program, the ARO Young Investigator Program and the Department of Defense.

As stated earlier, in the network coding setting, the rate allocation problem is an LP, and can be solved using centralized solvers. In addition, primal-dual and gradient-descent based techniques can be used to solve LP's efficiently over distributed networks. In this paper, we use a BP based mechanism for determining an (approximate) optimal solution to this LP. This work is unique in the following respects:

a. The BP solution is inherently lower in complexity than its other counterparts [10] and thus, when it converges, presents a more efficient mechanism for resource allocation than simplex/interior-point based LP solvers.

b. It is distributed in nature requiring local transfer of messages among nodes in the network. As the network topology evolves over time, this enables the discovery of a new rate allocation strategy efficiently in a distributed fashion.

c. Finally, the algorithm presented in this work is a non-trivial generalization of traditional BP algorithms for ILP's. Specifically, this work generalizes the traditional "sum-product" algorithm for BP's to an "integrate-product" algorithm. This new BP algorithm may prove useful in solving a much larger class of LP problems beyond rate allocation.

The rest of this paper is organized as follows: the next section describes the network mode. Two BP based rate allocation algorithms are presented in Section III. Finally, simulation results are presented in Section IV, and the discussion of the obtained results follows in Section V. This paper concludes with Section VII.

II. Network Model

The system of interest is a two-hop network as depicted in Figure 1. The network consists of one source node (denoted as T), one intermediate node S and a bipartite graph with two classes of set of nodes $A_1 = \{U_i\}$ and $A_2 = \{V_a\}$, consisting of N and M nodes, respectively. As a motivating example for our topology, we can say that nodes U_i's are local base stations, node S is a data sender (such as a satellite) communicating through the wireless medium with the local base stations, and nodes V_a's are local users. As users corresponding to V_a's move or channel conditions change, the

978-1-4244-3938-6/09/$25.00 (c)2009 IEEE

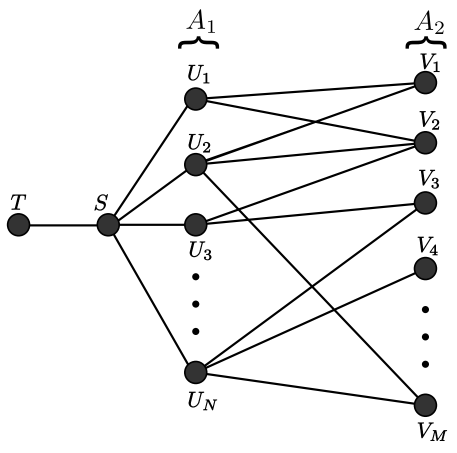

Fig. 1. Network Model

bipartite topology may change. Nodes U_i's perform broadcasting to all nodes in their local area. We are interested in optimizing the cost over all wireless links in the network. Network state (its bipartite topology) evolves in a quasi-static manner. In other words, we assume that its bipartite topology is in a state lasting for an extended period of time, and it subsequently changes to another bipartite topology and so on. Thus, the bipartite topology is unknown to nodes in the network. However, in a particular time period, we assume that each node knows its own neighbors. In our network model, we restrict each link to have a capacity (maximum rate of transmission) of one.[2]

We desire to multicast over the network configuration illustrated in Figure 1, meaning that each destination node V_i in A_2 receives identical information from the single source T. To this end, we employ linear network coding as encoding strategy at each node in this network. It is well known that, for multicasting, linear network coding is sufficient for achieving network capacity [3].

A glance at the network in Figure 1 indicates that the min-cut capacity from the source to each destination is one. Thus, for simplicity, we consider the cost function to be optimized in this paper to equal the sum of rates allocated to each link in the network. Note again that this cost function is chosen both for practical relevance and simplicity, and it in no way constrains the algorithms derived in the paper.

Our ultimate aims are to develop algorithms that assign rates to each link in a distributed manner such that they satisfy the following two properties:

1) Each destination node receives data at the multicast capacity of the network (one).
2) The overall cost incurred (sum of rates in the network) is minimized.

We further assume that there are local feedback links available so as to enable message-passing between adjacent nodes. Those messages are exchanged each time the network configuration changes to determine a new set of rates per link in the network.

[2]This is for simplicity only. Our algorithms generalize in a straightforward manner for more general capacity constraints.

III. Distributed Algorithms for Rate Allocation

In this section, we present two distributed algorithms for the rate allocation problem of the network described in Section II. Both algorithms are based on BP which is an iterative statistical inference technique widely used for constraint satisfaction problems. To match the terminology used in the existing framework of BP, the nodes in the set A_1 and A_2 will be called "variable nodes" and "constraint nodes," respectively. $\mathcal{N}(U_i)$ ($\mathcal{N}(V_a)$) denotes the neighbors of the node $U_i \in A_1$ ($V_a \in A_2$). Also, $m^{(l)}_{U_i \to V_a}(x)$ ($M^{(l)}_{V_a \to U_i}(x)$) denotes the message transferred from a variable node U_i (a constraint node V_a) to a constraint node V_a (a variable node U_i) in the lth iteration, given the assumption that the rate value allocated to U_i by the intermediate node S is x.

A. Algorithm I

This algorithm dictates that each message be a valid probability distribution. For the initial outgoing message from each variable node, any probability distribution with a "peak" at zero is a viable choice. For simplicity of representation, we set the outgoing message from each variable node to equal a Gaussian $f(x) = \frac{1}{Z}\exp(-\beta x^2)$ over possible rate values x, where Z is a normalization factor (also referred to as the partition function in statistical physics), and β is a constant that determines the resulting algorithm's speed of convergence (also referred to as temperature in statistical physics). Although the range of valid rate values is only $[0, 1]$, the initial message defined as above (over the range of $(-\infty, \infty)$) is empirically shown to work well and also facilitates the representation.

1. Initialization: Set the outgoing message from each variable node to a Gaussian function of the form $f(x) = \frac{1}{Z}\exp(-\beta x^2)$.

2. Constraint node updates: The outgoing message from a constraint node V_a to one of its adjacent variable node U_i is updated using the following equation:

$$M^{(l)}_{V_a \to U_i}(x) \tag{1}$$
$$= \frac{1}{Z_1}\int_{\mathbf{u}_i} \mathbb{I}\left(\sum_{k \neq i} u_k \geq 1 - x\right) \prod_{\substack{k \neq i \\ U_k \in \mathcal{N}(V_a)}} m^{(l)}_{U_k \to V_a}(u_k)\, d\mathbf{u}_i,$$

where $\mathbf{u}_i \triangleq (u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_{|\mathcal{N}(V_a)|})$ is a vector of rate values assumed by other adjacent variable nodes of V_a, and Z_1 is an appropriate normalization factor. Also, $\mathbb{I}$ is the indicator function whose output is one if the input constraint is satisfied, otherwise zero.

3. Variable node updates: The outgoing message from a constraint node V_a to one of its adjacent variable node U_i is updated using the following equation:

$$m^{(l+1)}_{U_i \to V_a}(x) = \frac{1}{Z_2} \prod_{\substack{b \neq a \\ V_b \in \mathcal{N}(U_i)}} M^{(l)}_{V_b \to U_i}(x)\exp(-\beta|x|), \tag{2}$$

where Z_2 is an appropriate normalization factor.

4. Belief calculation: Repeat Step 2 and Step 3 until the following (marginal) probability distribution converges to a fixed distribution:

$$p^{(l+1)}_{U_i}(x) = \frac{1}{Z_3} \prod_{V_a \in \mathcal{N}(U_i)} M^{(l)}_{V_a \to U_i}(x) \exp(-\beta|x|), \quad (3)$$

where Z_3 is an appropriate normalization factor.

B. Algorithm II

In this algorithm, each message is assumed to be from a Gaussian distribution.[3] Since the mean and the variance are sufficient statistics for the Gaussian probability distribution, those parameters are transferred back and forth as a (vector) message. Let $f(x; \mu, \sigma^2)$ denote the pdf of a Gaussian distribution with the mean μ and the variance σ^2 and let a function $Q(x; \mu, \sigma^2)$ be defined as

$$Q(x; \mu, \sigma^2) = \int_x^\infty f(z; \mu, \sigma^2)\, dz. \quad (4)$$

Also, for a pdf $f(x)$, let we define a (vector) functional $\text{MV}\left(f(x)\right) = \left(\mathbb{E}_f[x]\ ,\ \mathbb{E}_f[x^2] - \mathbb{E}_f[x]^2\right)$, i.e. the two dimensional vector consisting of the mean and the variance of the random variable associated with $f(x)$.

1. Initialization: Set the outgoing message of each variable node to the vector $(\mu^{(0)}_{U_i \to V_a}, \sigma^{2(0)}_{U_i \to V_a}) = (0, \frac{1}{2\beta})$, where β is a constant determining convergence speed as in Algorithm I.

2. Constraint node updates: The outgoing message from a constraint node V_a to one of its adjacent variable node U_i is updated using the following equation:

$$\left(\mu^{(l)}_{V_a \to U_i}, \sigma^{2(l)}_{V_a \to U_i}\right) = \sum_{\substack{k \neq i \\ U_k \in \mathcal{N}(V_a)}} m^{(l)}_{U_k \to V_a}. \quad (5)$$

3. Variable node updates: The outgoing message from a constraint node V_a to one of its adjacent variable node U_i is updated using the following equation:

$$m^{(l+1)}_{U_i \to V_a} = \text{MV}\left(g^{(l+1)}_{U_i \to V_a}(x)\right), \quad (6)$$

where

$$g^{(l+1)}_{U_i \to V_a}(x) = \frac{1}{Z_4} \prod_{\substack{b \neq a \\ V_b \in \mathcal{N}(U_i)}} Q(x; \mu^{(l)}_{V_b \to U_i}, \sigma^{2(l)}_{V_b \to U_i}) \exp(-\beta x^2),$$

and Z_4 is an appropriate normalization factor.

4. Belief calculation: Repeat Step 2 and Step 3 until the following (marginal) probability distribution converges to a fixed distribution:

$$f^{(l+1)}_{U_i}(x) = f(x; \mu^{(l+1)}_{U_i}, \sigma^{2(l+1)}_{U_i}), \quad (7)$$

where

$$\left(\mu^{(l+1)}_{U_i}, \sigma^{2(l+1)}_{U_i}\right) = \text{MV}\left(g^{(l+1)}_{U_i}(x)\right),$$

$$g^{(l+1)}_{U_i}(x) = \frac{1}{Z_5} \prod_{V_b \in \mathcal{N}(U_i)} Q(x; \mu^{(l)}_{V_b \to U_i}, \sigma^{2(l)}_{V_b \to U_i}) \exp(-\beta x^2),$$

[3]Note that this is an approximation, and the real distribution is not necessarily Gaussian.

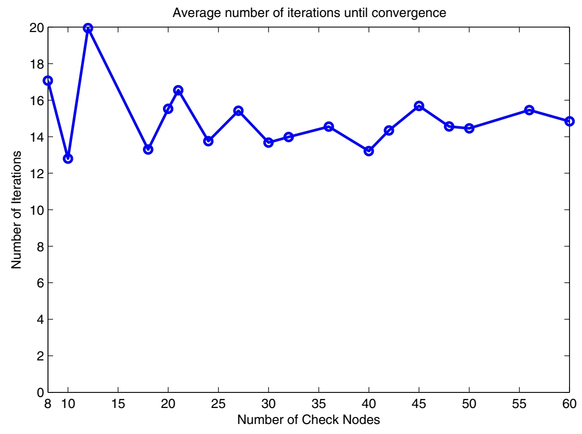

Fig. 2. Average number of iterations taken by Algorithm I until convergence

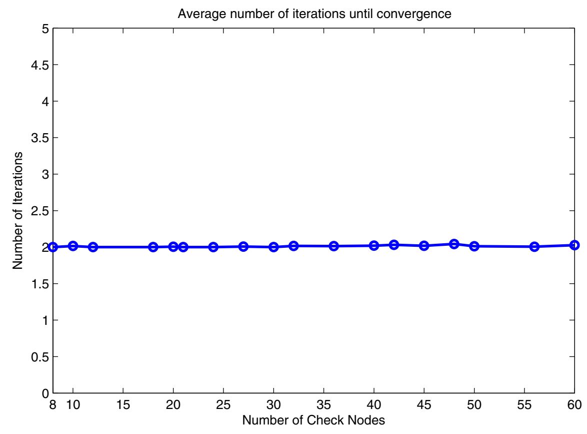

Fig. 3. Average number of iterations taken by Algorithm II until convergence

and Z_5 is an appropriate normalization factor.

IV. Simulation Results

In this section, we present simulation results for the two algorithms described in Section III. Incidence matrices of dimensions $\{7 \times 8, 8 \times 10, 10 \times 12, 15 \times 18, 17 \times 20, 18 \times 21, 20 \times 24, 23 \times 27, 25 \times 30, 27 \times 32, 30 \times 36, 33 \times 40, 35 \times 42, 38 \times 45, 40 \times 48, 42 \times 50, 47 \times 56, 50 \times 60\}$ were simulated for both algorithms. The matrices were generated so that their associated graphs do not contain cycles of length four for smaller dimensions and of length up to 10 for larger dimensions using cycle removal techniques introduced in LDPC code generation literature [4]. The degree of variable nodes was chosen to be three, and the degrees of constraint nodes were chosen randomly so that no degree-one constraint node happens which can be trivially removed for reducing the graph. The probability density function resolution was set so that a total range of $[-1, 3]$ is represented with 1024 points, instead of $[-\infty, \infty]$ for ease of computation. The algorithms were set to stop when there is no significant change in the distribution obtained in Step 4. Figure 10 illustrates an instance for the evolution of the distribution of one variable node with an

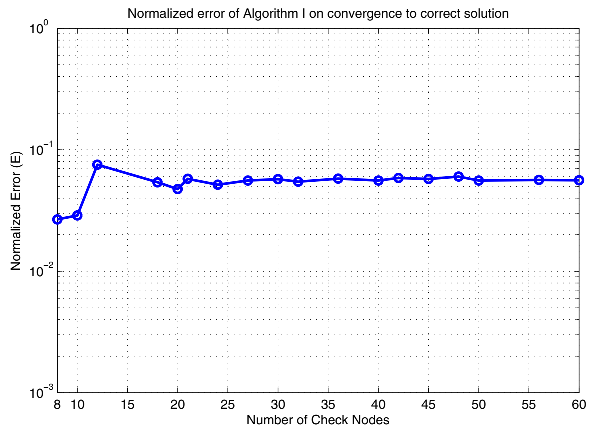

Fig. 4. Normalized error of Algorithm I on convergence to the correct solution

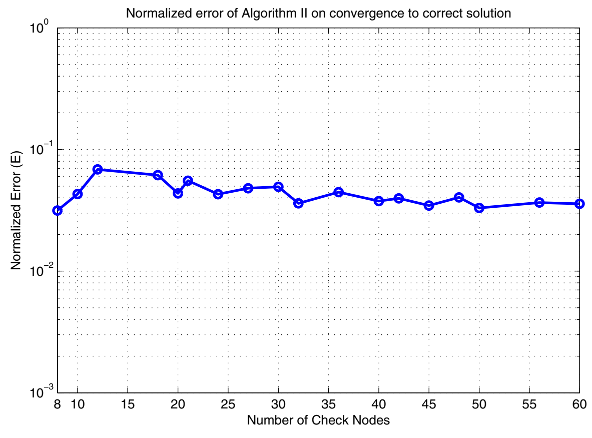

Fig. 7. Normalized error of Algorithm II on convergence to the correct solution

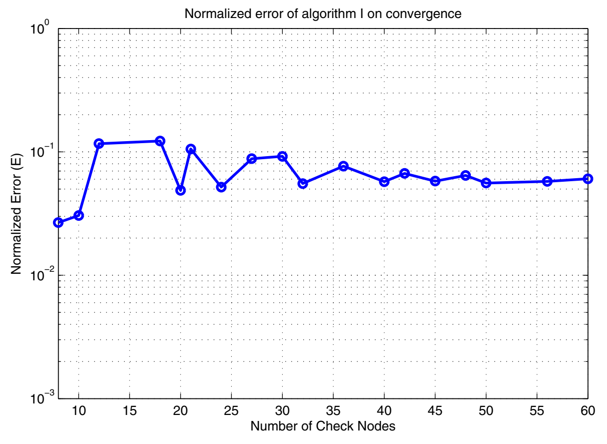

Fig. 5. Normalized error of Algorithm I on convergence

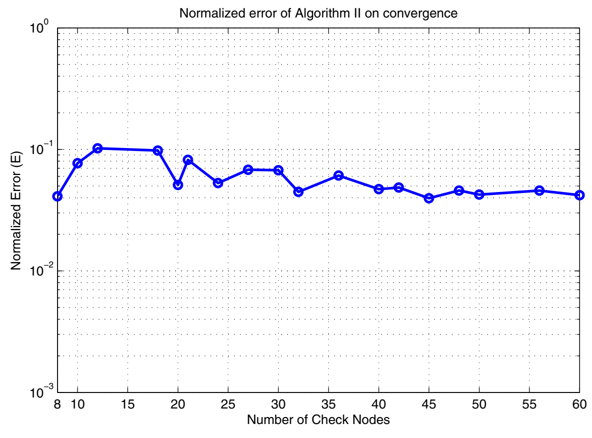

Fig. 8. Normalized error of Algorithm II on convergence

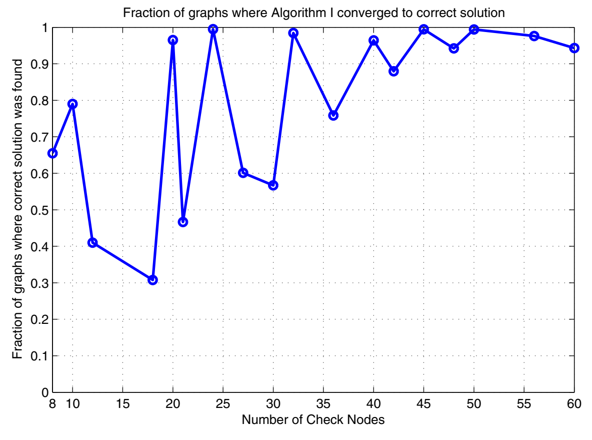

Fig. 6. Fraction of graphs where Algorithm I converged to the correct solution

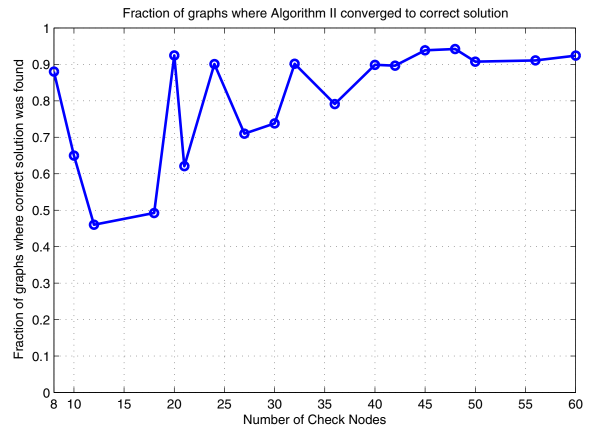

Fig. 9. Fraction of graphs where Algorithm II converged to the correct solution

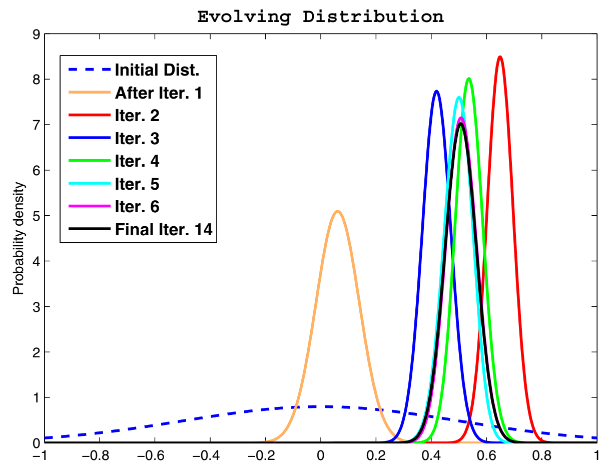

Fig. 10. An example of the evolution of probability distribution of a variable node

50×60 incidence matrix which converged in 14 iterations. We see from this figure that the peak of the distribution goes from zero to the region close to the solution, moves around the solution and finally converges to the solution.

In Figures 2, 4, 6, and 5, we present, for Algorithm I, the average number of iterations until convergence, the normalized error between the true value and the algorithm's output when the algorithm converges to the correct solution, the normalized error when the algorithm converges to a fixed point, and the fraction of graphs in the ensemble where the algorithm converges to the correct solution, respectively. The horizontal axis represents classes of matrices by the number of columns of matrices in the class. From Figure 2, we see that Algorithm I converges in around 15 iterations for most cases. The normalized error is defined by

$$e(\mathbf{u}_{\mathrm{BP}}) = \frac{\|\mathbf{u}_{\mathrm{BP}} - \mathbf{u}_{\mathrm{LP}}\|_1}{\|\mathbf{u}_{\mathrm{LP}}\|_1},$$

where $\mathbf{u}_{\mathrm{BP}}$ is the output of the proposed algorithm, and $\mathbf{u}_{\mathrm{LP}}$ is the solution of original formulation. The difference between the normalized error shown in Figure 4 and Figure 6 shows that there exists a collection of "ill-conditioned" matrices where Algorithm I fails to find the correct solution. Also, we see from Figure 5 that, for smaller dimension of incidence matrices, it is more likely that those "ill-conditioned" matrices happen.

In Figures 3, 7, 8, and 9, the average number of iterations until convergence, the normalized error when the algorithm converges to the correct solution, the normalized error when the algorithm converges to a fixed point, and the fraction of graphs in the ensemble where the algorithm converges to the correct solution are respectively depicted for Algorithm II. There are several interesting observations about Algorithms II as follows: i) Algorithm II converges in 2 iterations in most cases, while the number of iterations for Algorithm I varies from 12 to 20. ii) the normalized error of (correct) solutions obtained by Algorithm II is smaller than those obtained by Algorithm 1. iii) the fluctuation in the fraction of "ill-conditioned" matrices along the dimension is smaller than that for Algorithm I. Therefore, we see that Algorithm II is more robust in a randomly chosen graph.

V. Discussion

In this section, we explain the relationship between the algorithms derived in Section III and the LP that defines the original problem. The LP (after a few simple manipulations) can be expressed as:

$$\begin{aligned} &\min_{\{x_{U_i}\}} \quad \sum_{i=1}^{N} x_{U_i} \\ &\text{subject to} \\ \text{Constraint } V_a :& \sum_{U_k \in \mathcal{N}(V_a)} x_{U_k} \geq 1 \text{ for } 1 \leq a \leq M, \\ & 0 \leq x_{U_i} \leq 1 \text{ for } 1 \leq i \leq N, \end{aligned} \tag{8}$$

which can be written in terms of a an standard LP:

$$\begin{aligned} &\min_{\mathrm{x}} \quad \mathbb{1}_N^T \mathrm{x} \\ &\text{subject to} \\ &\mathbb{A}\mathrm{x} \geq \mathbb{1}_M, \end{aligned} \tag{9}$$

where x_{U_i} denotes the rate value allocated to node $U_i \in A_1$, $\mathrm{x} = [x_1, \ldots, x_{U_N}]^T$, $\mathbb{1}_k$ is the k-dimensional all-one column vector, and $\mathbb{A}$ is the $M \times N$ incidence matrix where the entry at (i,j) is one if there is an edge between the node $V_i \in A_2$ and the node $U_j \in A_1$; otherwise the entry is zero.

Since there is an explicit constraint that all x_{U_i}'s be non-negative, we can relax this constraint by changing the cost function to

$$\sum_{i=1}^{N} |x_{U_i}|. \tag{10}$$

Also, let us define a function that is associated with each constraint $V_a \in A_2$ as follows:

$$\mathbb{I}\left(\sum_{U_k \in \mathcal{N}(V_a)} x_{U_k} \geq 1\right), \tag{11}$$

where $\mathbb{I}(\cdot)$ is the indicator function. In other words, the output value of this function is one if the variable nodes U_i in the $\mathcal{N}(V_a)$, otherwise zero. Using (10) and (11), the LP problem can be converted to the following unconstrained optimization problem:

$$\max_{\{x_{U_k}\}} \prod_{a=1}^{M} \mathbb{I}\left(\sum_{U_k \in \mathcal{N}(V_a)} x_{U_k} \geq 1\right) \exp\left(-\beta \sum_{k=1}^{N} |x_{U_k}|\right), \tag{12}$$

where β is a nonnegative parameter that involves the convergence speed. The optimization problem like (12) can be solved using BP based message-passing techniques [9].

The messages defined in the Section III are updated according to the rules which are based on the probability of the event that each variable node takes a specific value at each time instant. Let X_{U_i} denote a (continuous) random variable associated with the rate value x_{U_i}. Here, there is an important underlying assumption: All messages on different edges in the network are independent. Thus, each type of messages has its own probabilistic interpretation: The message $M^{(l)}_{V_a \to U_i}(x)$ can be interpreted as the probability density of X_{U_i} being forced to take a value x in order for the constraint V_a to be satisfied. Subsequently, the message $m^{(l)}_{U_i \to V_a}(x)$ can be interpreted as the probability density of X_{U_i} being equal to x in order to satisfy all adjacent constraints when the constraint V_a is absent from the network, i.e. a priori knowledge for the probability density of X_{U_i} provided by other adjacent constraint nodes. Thus, the belief calculated in the algorithms corresponds to the marginal probability density of each variable node. Also, the probability density of X_{U_i} being x in the event that the constraint V_a is satisfied is straightforwardly calculated as:

$$
\begin{aligned}
&f^{(l)}_{U_i}(x| \sum_{U_k \in \mathcal{N}(V_a)} X_{U_k} \geq 1, X_{U_k} = u_k) \\
&= \frac{1}{Z_1} \int_{\mathbf{u}_i} \mathbb{I}\left(\sum_{k \neq i} u_k \geq 1 - x \right) \prod_{\substack{k \neq i \\ U_k \in \mathcal{N}(V_a)}} f^{(l)}_{U_k}(u_k)\, d\mathbf{u}_i, \quad (13)
\end{aligned}
$$

where $f^{(l)}_{U_k}(u_k)$ is a priori probability density of X_{U_k}, which also corresponds to the message $m^{(l)}_{U_i \to V_a}(x)$. Here, the distributions at variable nodes are assumed to be independent. Since (13) corresponds to the definition of the message $M^{(l)}_{V_a \to U_i}(x)$, this message takes such form as (1). Moreover, the message $m^{(l)}_{U_i \to V_a}(x)$ is calculated by taking the product of probability density of X_{U_i} transferred from other constraint nodes in the sense of a prior knowledge. These update rules form Algorithm I. Although Algorithm I basically uses the probability densities as its messages, the algorithm can be simplified by a further assumption that the message distribution be Gaussian-like with a (minor) change in the objective function to:

$$
\sum_{i=1}^{N} x^2_{U_i}. \quad (14)
$$

Then, the corresponding unconstrained optimization problem is given as

$$
\max_{\{x_{U_k}\}} \prod_{a=1}^{M} \mathbb{I}\left(\sum_{U_k \in \mathcal{N}(V_a)} x_{U_k} \geq 1 \right) \exp\left(-\beta \sum_{k=1}^{N} x^2_{U_k} \right). \quad (15)
$$

Note that the "local potential" term for X_{U_i} is a Gaussian function, and the constraint function basically involves a series of convolution operations followed by the calculation of Gaussian integral. Therefore, we can approximate messages transferred along the network to be Gaussian functions. Since, as mentioned in the previous section, the mean and the variance are sufficient statistics of Gaussian probability density under the approximation, the same procedure as Algorithm I can be performed by transferring only two parameters. Hence, Algorithm II is given as such.

VI. Complexity analysis

Here, we discuss the complexity of the proposed algorithms. Let L denote the number of points for the FFT operation to perform a convolution operation of two message distributions. For Algorithm I, a single edge processing at the constraint node update for a degree-d_c constraint node involves $d_c - 2$ convolutions and one integration, which are computationally equivalent to $d_c - 1$ FFT's and $d_c - 2$ products of message distributions. Since there are md_c edges connected to constraint nodes, the complexity for constraint node updates is $O(md_c^2 L \log L)$. Likewise, the complexity for variable node updates of degree d_v is $O(nd_v^2 L)$. Therefore, the overall complexity of Algorithm I is $O(nd_v^2 L \log L)$.

For Algorithm II, the constraint node updates are simple because all operations can be replaced by addition operations. Since one integration and $2(d_c - 2)$ additions are required for a single edge processing, the number of addition required for constraint node updates is $O(md_c(d_c + L))$. If we use a lookup table for $Q(x; \mu, \sigma^2)$, the number of additions reduces to $O(md_c^2)$. Since variable node updates involve direct calculation of the first and second moment of message distributions, the overall number of multiplications and additions are $O(nd_v^2 L)$ and $O(nd_v L)$, respectively. Therefore, the overall complexity of Algorithm II is $O(nd_v^2 L)$.

VII. Conclusion

In this work, we presented two distributed algorithms for the rate allocation problem over a coded multicast network. We provided simulation results which illustrates the convergence rate and correctness of these algorithms.

References

[1] M. Bayati, C. Borgs, J. Chayes, and R. Zecchina. Belief-propagation for weighted b-matchings on arbitrary graphs and its relation to linear programs with integer solutions. *Available at http://arxiv.org/abs/0709.1190*, Feb. 2008.

[2] D. Bertsekas and R. Gallager. *Data networks*. Prentice Hall, 1992.

[3] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Trans. on Info. Theory*, 52(10):4413–4430, Oct. 2006.

[4] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold. Regular and irregular progressive edge-growth tanner graphs. *IEEE Trans. Inform. Theory*, 51(1):386–398, Jan. 2005.

[5] J.M. Jaffe. Bottleneck flow control. *IEEE Trans. on Comm.*, 29(7):954–962, Jul. 1981.

[6] M. Kim, M. Médard, V. Aggarwal, and U. M. O'Reilly. On the coding-link cost tradeoff in multicast network coding. *Milit. Comm. Conf. (MILCOM)*, pages 1–7, Oct. 2007.

[7] D. S. Lun, M. Médard, T. Ho, and R. Koetter. Network coding with a cost criterion. *Int. Symp. on Info. Theory and its Appl. (ISITA)*, pages 1232–1237, Oct. 2004.

[8] J. Mosely. *Asynchronous Distributed Flow Control Algorithms*. PhD thesis, Dept. of Elec. Eng. and Comp. Sci., MIT, Cambridge, Mass, Jun. 1984.

[9] S. Sanghavi, D. Shah, and A. Willsky. Message-passing for maximum weight independent set. *Available at http://arxiv.org/abs/0807.5091*, Jul. 2008.

[10] N. Sommer, M. Feder, and O. Shalvi. Low-density lattice codes. *IEEE Trans. Inform. Theory*, 54(4):1561–1585, Apr. 2008.