A Joint Routing and Coding Approach to Linear Network Coding

Mostafa El-Khamy [♢,*]
[♢] Electrical Engineering Department, Alexandria University, Alexandria 21544, Egypt
[*]Egypt-Japan University of Science and Technology (E-JUST)
Email: mostafa@systems.caltech.edu

***Abstract*—Adopting a cross-layer approach, in this paper we propose an algorithm for joint routing and network coding. The proposed algorithm jointly assigns routes and designs linear network codes over finite fields to achieve the capacity of the network. The algorithm has a dynamic programming approach where a cost function is used to assign weights to all edges in the network. The cheapest flow is chosen subject to certain encoding constraints in order to achieve the network capacity with network coding while minimizing the network complexity. The effectiveness of the algorithm is demonstrated through carefully chosen examples. We show that the constraints imposed by the joint routing and coding algorithm are necessary for successful decoding at the sinks, and their violation can lead to a failure in achieving the network capacity or an increase in the number of encoding nodes.**

***Index Terms*—network coding; finite fields; linear coding; routing; networks; capacity, graph theory**

I. Introduction

Network coding was coined by Alshwede *et al.* where the authors demonstrated on the *butterfly network* the advantage of coding at the network nodes over conventional store-and-forward [1]. The rate of information flow in point-to-point communication networks is bounded by the max-flow min-cut theorem which states that the maximum flow between two nodes is equal to the value of the minimum cut between the two nodes [2]. Li *et al.* defined a linear code multicast for an acyclic network and proved that linear network codes can achieve the max-flow bound [3]. Koetter and Medard showed that for a solvable multicast network with one source of rate C and N sinks, there exists a solution to the network coding problem in the finite field $\mathbf{F}_{2^m}$ with $m \leq \lceil \log_2(NC+1) \rceil$ [4]. Polynomial time algorithms for constructing linear network codes were addressed by Jaggi *et al.* [5] and later by Yeung *et. al* [6] for generic network codes. The advantage of network coding over routing in a randomized setting has been demonstrated for rectangular grid networks [7]. Using an experimental setup for a 20-node wireless networks, Katti *et al.* have shown that deploying network coding over routing has potential gains and can increase the throughput of wireless networks [8].

In a cross-layer approach, we propose an algorithm in this paper for joint routing and linear network coding in multicast networks. Our proposed algorithm jointly assigns flows and encoding operations to the network. The algorithm has a dynamic programming approach where a certain cost function is used to assign weights to all edges in the network. Our proposed algorithm assigns flows and linear network codes to define both the routing and coding operations at the network nodes. The goal of the algorithm is to achieve the network capacity while minimizing the coding complexity of the network, determined by the number of nodes deploying network codes.

II. System Model

An acyclic network (G, s, U, R) is defined as follows: G is a directed graph (V, E) where V is the set of nodes (vertices) in the network, E is the set of edges such that $e_k \triangleq (h_k, t_k) \in E$ is an edge with head $H(e_k) = h_k$ and tail $T(e_k) = t_k$. $S \subset V$ is the set of source nodes at which the transmitted information symbols are generated and input to the network. $U \subset V$ is the set of sink nodes such that all or a subset of the information symbols should be regenerated at each sink node $u \in U$. The capacity of the edges is defined by the set $R = \{R(e) : e \in E\}$, where $R(e)$ is the maximum number of symbols from an alphabet χ that can be sent over an edge e. If the source alphabet χ is the finite field of q^m elements, $\chi = \{0, 1, \cdots, q^m - 1\}$, then the network is called an $\mathbf{F}_{q^m}$ network. For any node $v \in V$, $\Gamma_I(v)$ and $\Gamma_O(v)$ will denote the set of input edges and the set of output edges connected to node v respectively. The in-degree and out-degree of node $v \in V$ will be denoted by $|\Gamma_I(v)|$ and $|\Gamma_O(v)|$ respectively, where $|\Gamma|$ denotes the cardinality of the set Γ. The labeling of the edges in the directed graph G follows an ancestral ordering such that $i < j$ if $h_i = t_j$. Similarly, the labeling of the nodes follows an ancestral ordering such that for any edge, the labeling of its head is less than that of its tail. A path from a source $s \in S$ to a sink $u \in U$ is an ordered set of the sequence of nodes traversed following edges from s to u: $P_{su} = \{s, p, \cdots, u\}$ where node p is an ancestor of u.

In this paper we assume each edge e has a unit capacity $R(e) = 1$, i.e. for an $\mathbf{F}_{q^m}$ network only one symbol in $\mathbf{F}_{q^m}$ can be transmitted over one edge per unit time. A link between any two nodes has an integer capacity, such that if the capacity of the link exceeds one symbol, then this link is represented by the corresponding number (equal to the link capacity) of unit-capacity edges in parallel. For simplicity, we assume all edges and nodes in the network to be of zero delay, although the proposed algorithm can be easily modified to incorporate non-zero delays. In case of a network with a single source s, let

$|\Gamma_I(s)| = C_s$ be the maximum rate at which the information symbols are generated at the source s. The ordering of the edges is initialized such that the edges $e_k \in \Gamma_I(s)$ are labeled by $k \in \{-1, -2, \ldots, -C_s\}$. The set of information symbols generated at the source at time slot t is $\Lambda^t \subset \mathbf{F}_{q^m}$ where $|\Lambda^t| \leq C_s$ and the information symbols are labeled by $\lambda_i^t \in \Lambda^t$, $i = 1, 2, \ldots, C_s$. Let $Y^t(e_j)$ denote the random process, or symbol, carried by an edge e_j at time slot t , i.e. the input edges at the source carry $Y^t(e_{-i}) = \lambda_i^t$.

In a linear $\mathbf{F}_{q^m}$ network, the node operations follow

$$Y^{t+1}(e_j) = \sum_{e_i \in \Gamma_I(h_j)} \beta_{i,j} Y^t(e_i),$$

where $i < j$ and $\beta_{i,j} \in \mathbf{F}_{q^m}$. Let $C_s(u) \leq C_s$ be the required rate of connection from the source node s to a sink node (user) $u \in U$, then $|\Gamma_O(u)| = C_s(u)$. The output edges from sink u are labeled $e_{u,i}, i = 1, 2, ..C_s(u)$.

***Definition** 1:* Assuming integer rates $C_s(u)$ and unit capacity edges, a *flow* $\mathcal{F}_s(u)$ from the source node s to a receiver node u is an assignment of $C_s(u)$ disjoint *subflows*;

$$\mathcal{F}_s(u) = \bigcup_{k=1}^{C_s(u)} \phi(\lambda_k, u),$$

such that a subflow $\phi(\lambda_k, u)$ is a path constructing a unit capacity flow for symbol λ_k from source s to sink u and $\bigcap_k \phi(\lambda_k, u) = \emptyset$, meaning that unit capacity flows (subflows) of different symbols to the same sink u are edge disjoint.

III. Proposed Joint Routing and Coding Algorithm

The essence of our algorithm is to jointly assign flows and encoding operations to a feasible network. The algorithm has a dynamic programming approach where a certain cost function is used to assign weights to all edges in the network. The flows and node encoding operations are chosen subject to satisfy certain constraints. The goal is to find a solution to the network coding problem that achieves the capacity, if a feasible solution exists, and simultaneously minimize the number of nodes with encoding operations.

For simplicity, when assigning costs to the network, it is assumed that all edges (links) are identical and have unit capacity, and links with higher capacity are modeled as multiple edges in parallel, as discussed earlier. In the following algorithm, a cost $w_i = \omega(e_i)$ is assigned to each edge e_i, where the weights are assigned inductively starting from the sinks.

***Algorithm** 1:* **Weight Assignment:**

1. a) $\omega(e_{u,i}) = 1;\ i = 1, 2, \ldots, C_s(u)\ \forall\ u \in U.$
   b) $\omega(e_i) = \frac{|\Gamma_O(u)|}{|\Gamma_I(u)|}\ \forall\ e_i \in \Gamma_I(u),\ u \in U.$
   c) $\omega(e_i) = \frac{\sum_{e_j \in \Gamma_O(v)} \omega(e_j)}{|\Gamma_I(v)|},\ \forall\ e_i \in \Gamma_I(v),\ v \in V \setminus U.$

The cost ω_i represents the average number of $\mathbf{F}_{q^m}$ symbols that e_i carries if its unit capacity constraint is relaxed and all incident edges to the same node carry the same number of symbols such that all U sinks receive their desired rate:

$$w_i = w_k \text{ if } \{e_i, e_k\} \in \Gamma_I(v).$$

When assigning weights to edges, the weights are maximized by assuming a worst case analysis when all sinks receive different symbols. Thus, the assigned weights depend only on the required rates at the sinks and not on the specific symbols to be received. Note that 1b agrees with 1c because from 1a all edges output from a sink have unit weight. The proposed cost function can be further modified to take into account other network parameters such as link delays and failure rates.

The following algorithm selects the order in which sinks are assigned flows for the data symbols transmitted from source s. This sink ordering is crucial to the network coding algorithm, since an arbitrary ordering may result in a non-optimal assignment of symbols to edges and thus non-optimal network-coding operations at the network nodes that fail to achieve the capacity of the network.

***Algorithm** 2:* **Flow Calculation and Sink Ordering:**

2. a) For each sink node u, find the set $\Phi_s(u)$ containing all possible flows $\mathcal{F}_s(u)$ of capacity $C_s(u)$.
   b) Calculate the weight of the flows in 2a by

$$\mathcal{W}\left(\mathcal{F}_s(u)\right) = \sum_{k=1}^{C_s(u)} \sum_{e \in \phi(k,u)} \omega(e).$$

   c) Calculate the minimum flow weight of sink u:

$$\mathcal{W}_{\min}(u) = \min_{\mathcal{F}_s(u) \in \Phi_s(u)} \mathcal{W}\left(\mathcal{F}_s(u)\right).$$

   d) Sort the sinks in descending order of their minimum flow weights $\mathcal{W}_{\min}(u)$.

It is worth noting that steps 2a$\cdots$2c can be done by a variant of Dijkstra's shortest path tree algorithm. One can argue that this ordering algorithm is optimal as it prevents assigning critical bottleneck links to flows of sinks which do not necessary need them. This ordering algorithm gives the highest priority to the sink node whose flows must pass through the network bottlenecks. The proposed weight assignment algorithm gives a larger weight to links that serve more network nodes and are thus a source of network bottlenecks to the network. The minimum flow weight of sink u is the minimum cost of transmitting all $C_s(u)$ symbols; if there exist two possible flows of capacity $C_s(u)$, then the flow with smaller cost (weight) will constitute of links of smaller weights, thus avoiding high-weight links which are the network bottlenecks. The sink node whose flow weight is $\max_{u \in U} \mathcal{W}_{\min}(u)$ is the sink whose cheapest flows pass through the largest number of bottleneck edges in the network. Thus the best one can do is to start assigning coded flows to this sink and then to the other sinks in descending order of their $\mathcal{W}_{\min}(u)$.

The *overlap weight* between two subflows ϕ_1 and ϕ_2 is defined to be the sum of the weights of the edges which are common to both paths forming the subflows:

$$\Omega(\phi_1, \phi_2) = \sum_{e \in \phi_1 \cap \phi_2} \omega(e).$$

The following algorithm explains how sink nodes are assigned coded flows which in turn determine the network coding

operations at network nodes. Dropping the time index t, let $\Lambda_u \subset \Lambda$ be the set of source symbols to be generated at sink u and $|\Lambda_u| = C_s(u)$.

Algorithm 3: **Coded Flow Assignment:**

3. $\forall u \in U$, assign flows $\phi(\lambda_i, u)$ for $\lambda_i \in \Lambda_u$, $i = 1, 2, \cdots, C_s(u)$, such that $\phi(\lambda_i, u)$ is the path with minimum weight:

$$\arg \min_{\phi(\lambda_i,u)} \sum_{e \in \phi(\lambda_i,u)} \omega(e)$$

that satisfies

a) No overlap with subflows to the same sink, if possible: $\phi(\lambda_i, u) \cap \phi(\lambda_k, u) = \emptyset$, for $k, = 1, 2, \cdots, i-1$.

b) Minimum overlap of $\phi(\lambda_i, u)$ with all previously assigned flows (to sinks including u) that carry other symbols $\lambda_j \neq \lambda_i$:

$$\arg \min_{\phi(\lambda_i,u)} \left(\Omega(\phi(\lambda_i, u), \phi(\lambda_j, u'))\right)$$

c) If there exists more than one flow satisfying step 3b then choose the flow $\phi(\lambda_i, u)$ that maximizes the overlap weight within flows, previously assigned to other sinks u' and carrying the same symbol:

$$\arg \max_{\phi(\lambda_i,u)} \left(\Omega(\phi(\lambda_i, u), \phi(\lambda_i, u'))\right).$$

This proposed coded flow assignment algorithm determines the path of each symbol from the source s to its destination u. The first condition follows from *Definition 1*. The proposed algorithm chooses the path that minimizes the cost of the flows, determined by the weight assignment algorithm, subject to a max-min criteria. It chooses the path that has maximum overlap with paths carrying the same symbol to other sinks while having minimum overlap with paths carrying other symbols to other sinks. This done with the goal of minimizing the number of nodes doing network coding. With the cost criteria set to edges and the ordering of sinks done by *Alg. 2*, this flow assignment algorithm will reserve important bottleneck links to other sinks which may be in critical need for them to achieve their min-cut max-flow capacity.

After assigning all coded flows, one or more subflows may be passing through a certain edge carrying the same or different symbols to different destinations. We define $\Psi(e)$ to be the union set of different symbols λ_i carried by all subflows passing through this edge. (If no subflows pass through e, then $\Psi(e) = \emptyset$.) For any node v, let $\Psi_I(v) = \bigcup_{e' \in \Gamma_I(v)} \Psi(e')$, then, by the above flow assignment algorithm, it is easy to show that $\Psi(e) \subset \Psi_I(v)$ for any $e \in \Gamma_O(v)$. This will be utilized in the following algorithm that assigns the coding operations at the network nodes and code the symbols to be sent on the edges.

The proposed network code vector assignment algorithm is applied to the network nodes sequentially from the sources to the destination. The first node to be assigned its network coding operation is the source nodes s: coded symbols $Y(e) \in \mathbf{F}_q$ are assigned to all edges $e \in \Gamma_O(s)$ and corresponding required coding functions are assigned to s. The next nodes to be assigned code vectors are those nodes v' such that all edges $e \in \Gamma_I(v')$ have already been assigned their network coded symbols $Y(e) \in \mathbf{F}_{q^m}$. For each node v, the union set of all coded symbols on its input edges will be denoted by $Y_I(v) = \bigcup_{e \in \Gamma_I(v)} Y(e)$ and has cardinality $|Y_I(v)|$.

Algorithm 4: **Network Code Assignment:**

4. Assign network coding (NC) operations at network node v according to the following cases:

a) IF $|Y_I(v)| = 1$, $Y_I(v) = \lambda$, then

- For all $e \in \Gamma_O(v)$ such that $\Psi(e) \neq \emptyset$, $Y(e) = \lambda$ and no NC is done at v

ELSE

b) IF $|\Gamma_O(v)| = 1$, $\Gamma_O(v) = \{e\}$, then

- $Y(e) = \sum_{\lambda_i \in \Psi(e)} \lambda_i$. IF $Y(e) \notin Y_I(v)$, NC is done at v, ELSE forward $Y(e)$.

ELSE

c) IF $|Y_I(v)| > 1$, $|\Gamma_O(v)| > 1$; $\forall e \in \Gamma_O(v)$

- IF $\Psi(e) = \emptyset$, no code symbol is sent on e.
- IF $|\Psi(e)| \geq 1$, then $Y(e) = \sum_{\lambda_i \in \Psi(e)} \lambda_i$:
  - i) IF $Y(e) \in Y_I(v)$, the symbol is forwarded and no NC is required.
  - ii) IF $Y(e) \notin Y_I(v)$, do NC: $Y(e)$ is a linear combination of elements in $Y_I(v)$.

This algorithm determines the coding operations to be done at the nodes based on the assignment of symbols to be carried by the edges determined by *Alg. 3*. Since the flow assignment algorithm guarantees maximum overlap between subflows carrying the same symbol and minimum overlap between subflows carrying different symbols, then the number of edges e having $|\Psi(e)| > 1$ will be minimal. It follows that the proposed joint routing and coding algorithm minimizes the number of network nodes with network coding operations.

IV. Illustrative Examples

In the following examples, we show that the conditions set by the proposed joint coding and routing algorithm are sufficient to finding a solution to the network coding problem while achieving the minimum-cut capacity with the minimum number of encoding nodes. We also show by counter-examples that these conditions are necessary and violating any of the rules or conditions set by the algorithm can lead to a non-optimal solution.

Example 1: In the network of Fig. 1, the source S broadcasts symbols a, b and c to both sinks, U_1 and U_2.

The network coding and routing algorithm is initialized by assigning weights to each edge in the network using *Alg. 1*. The edges incident to sinks U_1 and U_2 will have a weight of 1, since each sink has 3 incoming edges and should be able to regenerate the 3 symbols. The single edge incident to node 5 will be assigned a weight of 2, since node 5 has 2 outgoing edges, each of weight 1. Following the steps of *Alg. 1*, the weights are shown to the left of each edge in Fig. 1a. For example, each of the 3 edges incoming to node 3 will carry a

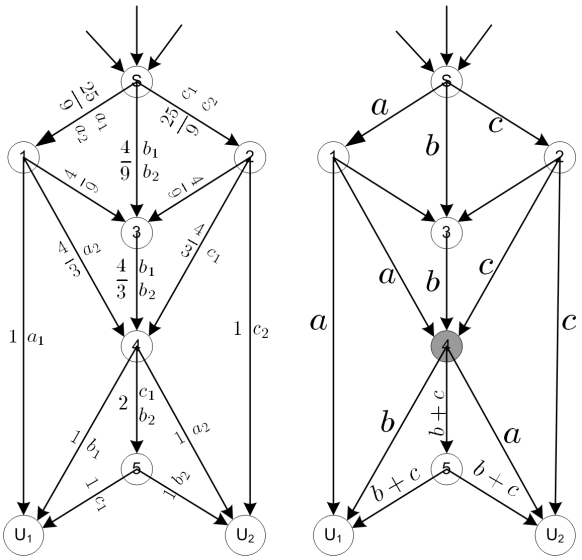


Fig. 1. Example 1: a) Routing and Coding, b) Coded Network

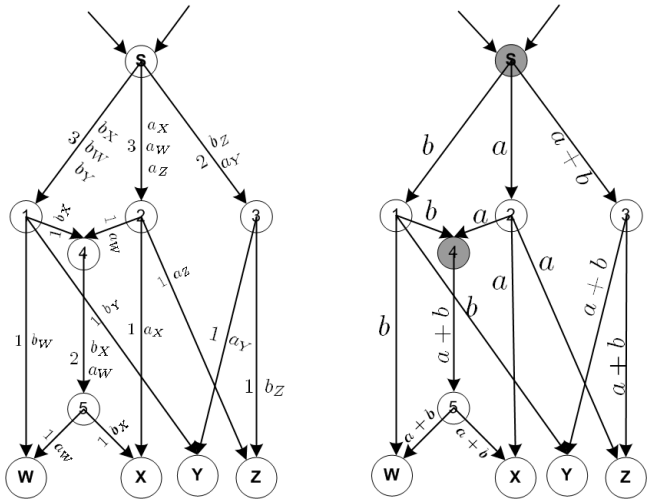


Fig. 2. Example 2: a) Routing and Coding, b) Coded Network

weight equal to the sum of the weights of the outgoing edges divided by the number of incoming edges,i.e. $\frac{1}{3}\frac{4}{3} = \frac{4}{9}$.

Since the network is symmetric with respect to U_1 and U_2, then *Alg. 2* will assign equal priority to both sinks. Suppose we start assigning flows to sink U_1, for the symbols b, a, and c, in that order (which can be chosen randomly). The flow of symbol b from S to U_1 will be labeled by b_1 and is assigned to the path with the smallest cost from S to U_1, $\{S, 3, 4, U_1\}$. Following *Alg. 3*, we assign edges to the flow a_1, which are chosen to construct the path of lowest cost that does not overlap with the flow b_1; $\{S, 1, U_1\}$. For c_1, the only available path that does not overlap with the flows a_1 and b_1 is $\{S, 2, 4, 5, U_1\}$. The flows passing through each edge are labeled to its right in in Fig. 1a, in the order they are assigned. We then proceed to assign flows to sink U_2 for symbols a, b, and c, in that order. The subflow a_2 should have minimum overlap with the subflows b_1 and c_1. Thus it should pass through edge $(S, 1)$, also to have maximum overlap with a_1. There are a number of such possible flows from S to U_2, e.g. $\{S, 1, 4, U_2\}$,$\{S, 1, 4, 5, U_2\}$, $\{S, 1, 3, 4, U_2\}$ and $\{S, 1, 3, 4, 5, U_2\}$. The first flow is chosen since it is the one with the minimum (zero) overlap with the flows b_1 and c_1. Similarly, the flow b_2 is chosen to be $\{S, 3, 4, 5, U_2\}$ to avoid overlap with subflows a_1, a_2 and c_2. The flow for symbol c to U_2 is chosen to have no overlap with the flows a_2 and b_2; $\{S, 2, U_2\}$. After assigning flows to all symbols to be sent to the sinks, the next and final step is to determine which symbols will actually be sent on the edges and the network coding operations at the network nodes. Following *Alg. 4*, it follows that only one node 4 will perform network coding where b and c are added and $b + c$ is sent on edge $(4, 5)$ and then forwarded to the sinks, as shown in Fig. 1b. It then follows that the sinks receive a, b and $b + c$ from which all source symbols can be retrieved. One can observe that no symbols have been carried by edges $(1, 3)$ and $(2, 3)$ and they can be considered redundant edges. One can also observe that the proposed algorithm performs both routing and coding in one shot and is successful in achieving the *min-cut capacity* of the network with only one coding operation.

Example 2: In the network of Fig. 2, the source broadcasts two symbols a and b to sinks W, X, Y and Z. The weights assigned to each edge using *Alg. 1* are labeled to the left of edges in Fig. 2a. The priority of the sinks is determined according to their minimum flow weight (step 2.c), where $\mathcal{W}_{\min}(W) = \mathcal{W}_{\min}(X) = 11$, $\mathcal{W}_{\min}(Y) = \mathcal{W}_{\min}(Z) = 7$. So the order of sinks to which flows for symbols a and b are assigned is chosen to be (X, W, Z, Y). We follow the same notation as in the previous example, where the subflow to sink X carrying symbol a is labeled a_X. Subflow a_X is assigned to the shortest path $\{S, 2, X\}$ and then $\{S, 1, 4, 5, X\}$ is assigned to b_X. The flow a_W is chosen according to the conditions set by *Alg. 3* to be $\{S, 2, 4, 5, W\}$ followed by assigning $\{S, 1, W\}$ for b_W. $\{S, 2, Z\}$ is chosen for the flow a_Z to have maximum overlap with other previously assigned flows carrying a and minimum overlap with those carrying b. Thus the only possible path for b_Z will be $\{S, 3, Z\}$. The subflow a_Y can be assigned to either $\{S, 3, Y\}$ or $\{S, 1, Y\}$. However, $\{S, 3, Y\}$ is chosen since it has a smaller overlap weight (2 compared to 3) with subflows carrying b. Consequently, b_Y is assigned to $\{S, 1, Y\}$. Using *Alg. 4*, the symbols to be transmitted on edges are shown and it becomes clear that both nodes S and 4 will be required to perform network codes. It is also clear that all sinks will be able to decode the transmitted data correctly since they receive $a + b$ and either a or b.

Counter-Example 1: In this counterexample, we use the same network of Example 2. We follow the same steps as in Example 2, except when assigning flows to Y. We violate condition 3b, and assign $\{S, 1, Y\}$ to a_Y followed by $\{S, 3, Y\}$ to b_Y. This is shown in Fig. 3a. By *Alg. 4*, both edges $(S, 1)$ and $(4, 5)$ will carry $a + b$ and only node S will be required to do network coding. However, it will follow that sink W will not receive the symbol $a + b$ only and will not be able to

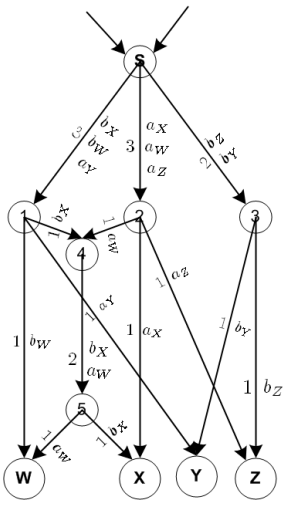
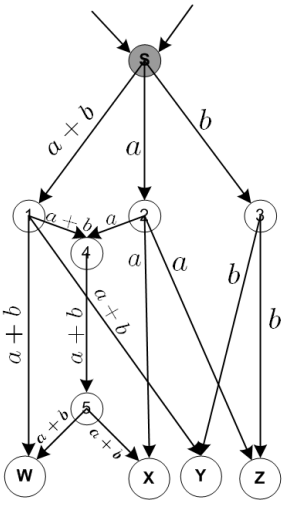


Fig. 3. Counter-Example 1: a) Routing and Coding, b) Coded Network



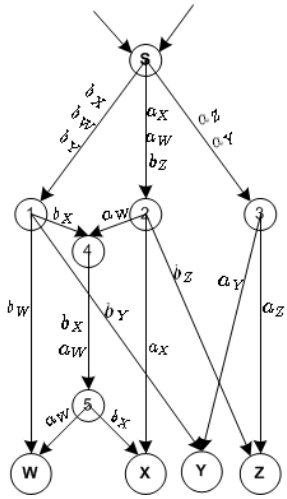
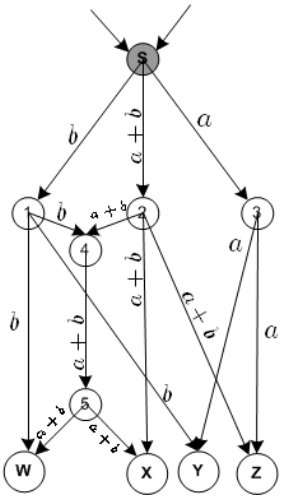


Fig. 4. Counter-Example 2: a) Routing and Coding, b) Coded Network



decode a or b, as shown in Fig. 3b. This confirms the following remark:

***Remark** 1: The minimum overlap rule of 3b is a necessary condition for achieving the network capacity with the joint routing and coding algorithm.*

Counter-Example 2: In this counterexample, we use the same network of Example 2. We follow the same steps as in Example 2, and assign flows to X and W. When assigning path to subflow a_Z, the possible paths are $\{S, 2, Z\}$ and $\{S, 3, Z\}$. For both paths, the overlap weight with subflows carrying b is zero. The overlap weight with subflows carrying a is 3 and zero respectively. By 3c one should choose $\{S, 2, Z\}$. In Fig. 4a, we violate this and choose $\{S, 3, Z\}$ instead. We then proceed by the algorithm and assign the flow for sink Y. The assigned flows and the corresponding transmitted coded symbols are shown in Fig. 4b. One can observe that this network code fails to transmit the symbols a and b to sink X. Thus we conclude the following remark:

***Remark** 2: The maximum overlap rule of 3c is a necessary condition for achieving the network capacity with the joint routing and coding algorithm.*

Counter-Example 3: In this counterexample, we show that if the condition of assigning subflows to the path with least cost (Step 3) is violated, then one can fail to find the optimum network code. We consider the network of Fig. 1. As in Example 1, we assign the flows $b_1, a_1, c_1, a_2, b_2, c_2$ in that order. We violate Step 3 when assigning flows to b_1 only, where we assign b_1 to $\{S, 1, 3, 4, 5, U_1\}$ whose cost is higher than the optimum path $\{S, 3, 4, U_1\}$ assigned in Example 1. We then proceed with the algorithm to assign the remaining flows. Fig. 5a shows the flows labeled on the edges in the order they are assigned. We note that the subflows available for c_1 either overlap with that of a_1 or b_1 as a result of violating the minimum weight rule in b_1, which also violates step 3.a. We assign c_1 to $\{S, 3, 4, U_1\}$ since it has the minimum overlap weight with subflows a_1 and b_1. We then proceed to assigning flows for U_2. Fig. 5b shows the resultant network code using

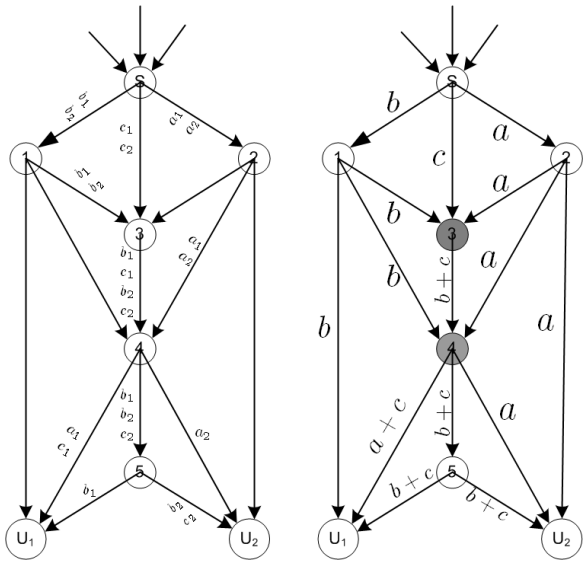


Fig. 5. Counter-Example 3: a) Routing and Coding, b) Coded Network

Alg. 4. We can observe that it will not be possible to decode all transmitted symbols at sink U_2, since it receives the symbols a and $b + c$ only. Comparing this to Example 1, we conclude the following remark:

***Remark** 3: When assigning subflows with the joint routing and coding algorithm, abiding by the minimum path-weight criterion described by Alg. 3, where the cost is calculated by Alg. 1, is a necessary condition for achieving the network capacity.*

Example 3: We consider the network of Fig. 6. Both sinks should receive symbols a and b from the source S. Using the weight criteria, the weight of sinks X and Y are $17/6$ and $47/6$ respectively. So we begin assigning subflows for symbols a and b to sink Y and then sink X. Following the proposed algorithm, the flows are labeled on Fig. 6a in the order they are assigned. Based on this, Alg. 4 is used to assign network codes at nodes and the transmitted symbols on edges. This is shown

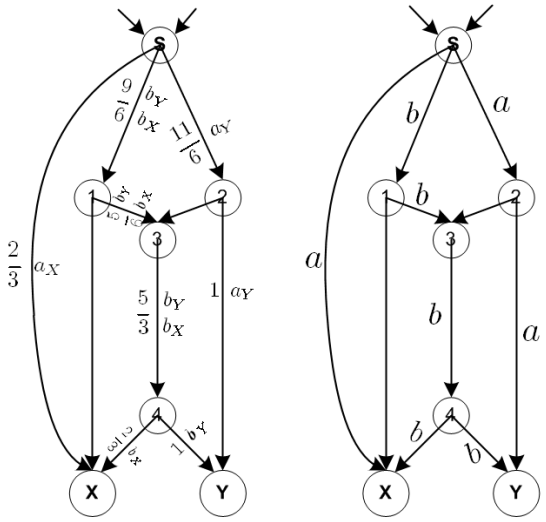


Fig. 6. Example 3: a) Routing and Coding, b) Coded Network

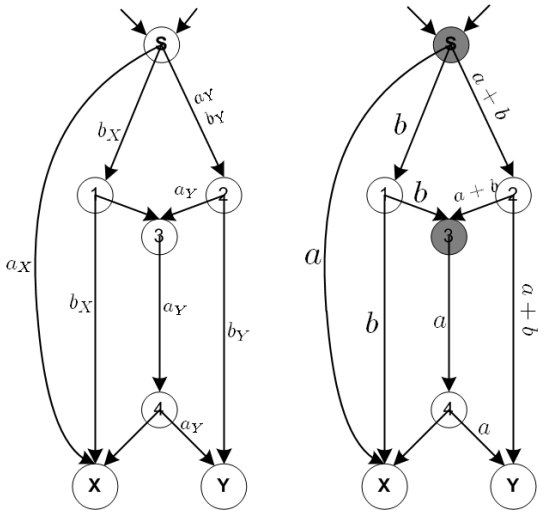


Fig. 7. Counter-Example 4: a) Routing and Coding, b) Coded Network

in Fig. 6b. It is worth noting that the proposed algorithm is successful in finding a valid assignment and no node will be required to form network coding

Counter-Example 4: We consider the same network as Example 3. We show that if one violates the minimum path-weight criterion of Alg. 3 when assigning flows, then one can fail to find the network code with the minimum number of encoding nodes. Similar to Example 3, we start by assigning the subflow a_Y. However, we assign it to the path of $\{S, 2, 3, 4, Y\}$, which has a higher cost than $\{S, 2, Y\}$. Remaining flows to b_Y will violate step 3a. We assign b_Y to $\{S, 2, Y\}$ since it is the path with minimum overlap with the subflow a_Y. We then proceed to assign flows to X according to the proposed algorithm as shown in Fig. 7a. The resultant coded network is shown in Fig. 7b. We observe that both sinks will be able to decode the two source symbols. However, this network coding assignment requires that two nodes form network coding operations which means more network complexity than that of Example 3. Thus we conclude the following remark.

***Remark** 4: When assigning subflows with the joint routing and coding algorithm, abiding by the minimum path-weight criterion described by Alg. 3, where the cost is calculated by Alg. 1, is a necessary condition for minimizing the number of coding nodes (minimizing the network coding complexity).*

Counter-Example 5: If we assign subflows to sinks of the network in Fig. 2 in the order Z, Y, X, W, one can show that network coding will fail and conclude the following remark:

***Remark** 5: Priority ordering of sinks according to Alg. 2 is a necessary condition for achieving the network capacity with the joint routing and coding algorithm.*

V. Conclusion

In this paper, we proposed a joint routing and coding algorithm for routing and linear network coding in multicast networks. Our algorithm is initialized by assigning weights to the links in the network to identify network bottlenecks. Consequently, the order of assigning flows to sinks is determined by giving more priority to sinks whose flows must utilize the network bottlenecks. Routing and coding of subflows for data symbols is jointly performed following certain constraints to avoid network bottlenecks and ensure successful decoding at the sinks with network coding. Through carefully chosen examples, we show that the conditions determined by our proposed algorithm are sufficient for successful routing and coding and achieving full network capacity. Through counter-examples, we show that the conditions set by our proposed joint routing and coding algorithm are also necessary to design a network code that achieves the full network capacity with the least network coding complexity.

Acknowledgment

This research was partially supported by the Egyptian National Telecommunications Regulatory Authority (NTRA). The author thanks M. Effros for inspiring discussions.

References

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *Information Theory, IEEE Transactions on*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] R. Yeung, *Information theory and network coding*. Springer Verlag, 2008.

[3] S. Li, R. Yeung, and N. Cai, "Linear network coding," *Information Theory, IEEE Transactions on*, vol. 49, no. 2, pp. 371–381, 2003.

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

[5] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *Information Theory, IEEE Transactions on*, vol. 51, no. 6, pp. 1973–1982, 2005.

[6] R. Yeung, S. Li, and N. Cai, *Network coding theory*. Now Pub, 2006.

[7] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE international symposium on information theory*. Citeseer, 2003, pp. 442–442.

[8] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2006, pp. 243–254.