Some Further Investigation on Maximum Throughput: Does Network Coding Really Help ?

Eric Gourdin, Yuhui Wang
France Télécom, Orange Labs, CORE/TPN/TRM
email: (name).(lastname)@orange.com

***Abstract*—Network coding has been shown to be the solution that allows to reach the theoretical maximum throughput in a capacitated telecommunication network [1]. It has also been shown to be a very appealing and practical alternative to routing-based approaches to send traffic from sources (servers) to terminals (clients) for many different applications.**

However, the initial theoretical claim of throughput benefit remains relatively unclear, mainly because the multicast throughput maximization problem is difficult to solve (it is closely related to the fractional Steiner tree packing problem which is NP-hard). In this paper, we show that these optimization problems are still tractable even for instances with a significant size (up to 50 nodes and 300 edges). We also propose and solve the multicast maximum throughput problem with an additional constraint on the number of multicast trees. We apply our algorithms on large sets of randomly generated instances, mainly based on bidirected graphs, because they are the most relevant to model fixed telecommunication infrastructures.

The main result of our intensive experimental study is that, in practice, network coding does not increase throughput compared to traditional multicast. Instances showing a throughput gain can only be generated somewhat artificially by imposing some structure or trying to maximize the throughput gap. However, when we limit the number of multicast trees, then, most of the times, very significant throughput gaps appeared. Since management constraints often impose on network administrators a very limited use of multicast trees, network coding appears clearly as a very nice alternative for delivering content to customers.

I. Introduction

Network coding has been first introduced as a nice solution allowing to reach a theoretical upper-bound on the throughput in multicast networks [1]. Given a capacitated network with a source node and a set of terminal nodes, this theoretical upper-bound refers to the global throughput that would be achieved if the stream between the source and each terminal node could use all the available resources (capacities), regardless of the other streams. The maximum throughput between a source and a destination can hence be obtained as the optimal solution of the well-known Maximum-Flow problem [2], [3].

Since then, network coding [4] has been considerably studied, both from the theoretical and the practical sides [5]. As a natural extension to the field of coding theory, several efficient ways to generate coding schemes and decoding algorithms have been proposed [6], [7], [8], [9]. Network coding applications have been proposed, and sometimes tested in various networks, in various places within the networks, ranging from the application layer to the physical layer, and for various applications [10], [11], [12].

Because of the original statement that network coding permits to reach the maximum possible throughput, a special emphasis has been laid on theoretical and empirical throughput evaluations, essentially focused on a comparison with standard multicast routing [13]. Recall that multicast protocols are designed for services where several users require the same content at the same time (live TV for instance). Multicast (one-to-many) is hence opposed to traditional unicast (one-to-one). The key feature of multicast protocols is to allow some well-chosen network nodes to replicate data towards several outgoing links in order to alleviate network load. Alternatively, network coding (associated with multicast or not) allows network nodes to combine several data received potentially from different incoming links towards one or several outgoing links. The famous *Butterfly* example is often cited to illustrate the benefit of network coding. We propose another example where the network coding throughput of 3 (Figure 1) is larger than the multicast throughput of 2.666 (Figure 2).

Many papers have been devoted to investigate this difference in throughput over different instances types and sets. A main result states that the relative gap in throughput is never larger than 2 together and some other theoretical bounds are proposed in [14]. When tackling the issue of comparing network coding with multicast throughput, one has to face optimization problems from graph theory. The first one, the maximum-flow problem, is very well-known and can be solved very efficiently (see, for instance, [15]). Solving a series of maximum-flow problems, one can derive the maximum network coding throughput. On the other-hand, computing the maximum multicast throughput boils down to consider the fractional Steiner tree packing problem, an NP-hard combinatorial problem [16], [17]. This problem can be related to another well-known NP-hard problem, namely the Steiner tree problem where one has to find a minimum cost subgraph interconnecting (or spanning) a given subset of nodes (called terminals). It is a common belief that these problems are almost impossible to solve in practice. However, using efficient Mixed-Integer Models [18], large instances of Steiner tree problems can be solved in a few seconds [19]. As a result, significant instances of the fractional Steiner tree packing problem can also be solved and thus, maximum multicast throughput can be evaluate on large sets of instances. To the best of our knowledge, all the previous empirical approaches to compare network coding and multicast throughput have either used approximation algorithms to solve the Steiner tree

packing problem, or have considered specific graphs instances where the resolution of the problem appeared to be more tractable. In [20], the authors consider the undirected case and claim that it is more general than the directed one. They also provide a Linear-Programming model using what they call *conceptual flows* to compute the network coding maximum throughput. Note that this model is nothing else than a very classical flow model that can trivially be derived from directed network coding models such as, for instance the one in [21]. Moreover, the authors in [20] use a standard simplex algorithm to solve a problem for which very much more efficient combinatorial methods are available (max-flow algorithms). They restrict their attention to uniform bi-partite graphs, whereas real-life network topologies can have very different structures. Finally, they use a brute force algorithm to solve the Steiner tree packing problem by enumerating all possible trees and then claim the problem cannot be solved even for medium size instances.

Since [20] seems to be the most comprehensive empirical study of network coding versus other routing paradigm throughput, we believe our paper will fill some major gap in this important field. Our contributions are the following:

- We propose efficient resolution schemes for the multicast maximum throughput problem (fractional Steiner tree packing problem) in the directed, bidirected (for each link, there is one arc in each direction) and undirected case. The chosen approach is a standard column generation algorithm where each column represents a Steiner tree. The pricing problems are hence Steiner tree problems for which we use a flow type model as in [18]. We also handle the maximum multicast throughput problem with a bounded number of trees, which, to the best of our knowledge, has not been addressed until now. For this problem, we propose an exact flow based model and a heuristic approach using tree variables.
- We somehow clarify the differences and relationships between directed, bidirected and undirected settings. We also claim that the bidirected case is quite relevant for telecommunication models because most fixed networks infrastructures have systematically two opposite links of same capacity.
- We provide numerical results based on extensive computational experiments (each set contains one hundred instances and the small topologies were checked with much larger number of instances). All instance are randomly generated without any predefined underlying structure and all results provided are the exact solutions of the related throughput problems.

Our main findings are:

- We confirm (but over a much larger scope of instances) what was announced in [20], namely that it is almost impossible to find instances exhibiting a non-zero gap between network coding and multicast throughput. In other words, the probability to pick a "butterfly-type" instance is almost zero.
- Concentrating on very small graphs (7 to 10 nodes), we have generated a very large number of instances and, in the smallest cases, even enumerated all possible instances. In this last setting, we were hence able to evaluate the probablity of picking up a topology with a non-zero gap, to be as low as 0.01%.
- Using again mixed-integer models, we could confirm numerically that the largest gap for uniform instances up to 7 nodes is 0.5.
- Finally, and this is not a surprise, we could evaluate the impact of the number of trees in the multicast solutions, and confirm that, when the number of trees is limited, there is often a huge gap between network coding and multicast throughput.

To summarize these results, we could say that, from a numerical point of view, it is wrong to claim that network coding allows to achieve an significant improvement in throughput. However, from a practical point of view, since telecommunication operators are reluctant, for obvious management complexity reasons, to handle large (and even sets of moderate size) sets of multicast trees, there is a very significant advantage that can be achieved in using network coding: network coding is much easier to deploy than multicast techniques, for a similar throughput when the number of trees to manage is large, and for a much better throughput when the number of tree is small. Of course, there are also many other interesting features offered by network coding (dynamicity, robustness, ...) which have been already largely promoted in the literature.

II. Compared coding and routing schemes

In this paper, our aim is to compare the throughput achieved while using different coding/routing paradigms:

- **Network Coding (NC)**: we use the result of [1] to compute the throughput achieved using network coding and multicast forwarding. Solving independently the sequence of maximum flow problems allows to obtain the value of the network coding throughput. The result obtained for a small graph G_1 is depicted in Figure 1: the instance has one source, three terminals, all arcs have a capacity of 1, except the arc (s, v_2) of capacity 2. three streams a, b and c (each one representing a volume of 1) are sent by the source node s. The node v_2, v_4 and v_5 perform coding on their input streams (for instance $a \oplus b$ means that stream a and b are coded together, resulting in a stream of volume 1). As a result, the terminal nodes receive each three different coded or uncoded streams, from which they can all decode the original streams a, b and c: the optimal network coding throughput is hence 3.
- **Multicast (MC)**: by Multicast, we mean here that several multicast trees can be set up between one source and all of its terminal, and the traffic is split appropriately among the chosen trees to reach the best possible throughput. The optimal throughput in G_1 is $8/3 \approx 2.666$, using 5 trees (see Figure 2). The original data are hence sliced

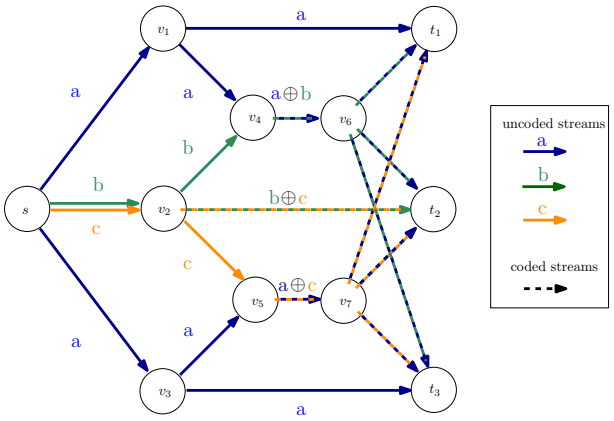

Fig. 1. Optimal solution with Network Coding (NC): throughput = 3, 3 streams a, b and c can be transmitted to each terminal.

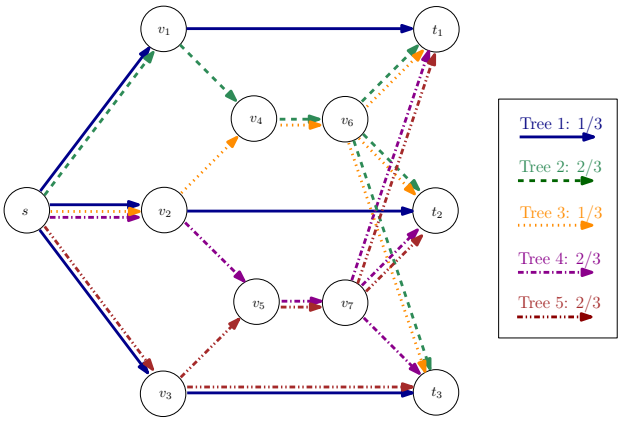

Fig. 2. Optimal solution with Multicast (MC): throughput = 8/3, but 5 different multicast trees are required.

into 5 streams, each one send on a different multicast tree.

- **Multicast with ℓ Trees (MC-ℓ)**: since the previous case is not often realistic (because network operator won't agree to handle many different multicast trees rooted at the same source), we also consider the case where the streams can only be routed over a limited number of multicast trees (up to ℓ and, in the numerical experiments, we will only consider the cases $\ell = 1, 2$ or 3). The optimal throughput in G_1 is 1 with one tree and 2 with two trees.

III. Brief survey of related optimization problems

The notion of maximum throughput is clearly defined when we consider a single stream between a source s and a destination t. The problem then reduces to the well-known maximum flow problem and the value of the maximum flow is equal to the value of the minimum $s-t$ cut [2], [3]. The strength of Network Coding is that the same principle can be applied when several streams (having a common source node s) must be routed concurrently in the graph [1]: more precisely, when the streams are coded in the network, then each stream can achieve the same throughput as if it would be the only stream present in the graph. Each individual throughput can then be computed again using a maximum flow algorithm. In other words, when Network Coding is used, there is no competition between the streams in using the network resources. However, without Network Coding, this is no longer true, and we then have to decide how the available resources (capacities) can be shared among the various streams. If R^k denotes the throughput achieved for the stream k, then the two most popular ways to define maximum throughput are:

$$\max \sum_{k \in K} R^k, \tag{1}$$

$$\max\{\lambda : \ R^k = \lambda d^k\}. \tag{2}$$

In (1), the sum of all individual throughputs is maximized whereas (2) corresponds to maxmin fairness.

In multicast networks, the routing protocols build and maintain a structure that interconnect all participants in a multicast session. For several reasons, including the fact that network operators want to minimize routing costs, these structures are almost always trees (a connected subgraph without cycles). A *spanning* tree is a tree that spans all nodes of the graph. Given an undirected graph $G_u = (V, E)$ and weight value w_e associated with the edges, a minimum spanning tree, is a spanning tree of the minimum weight. This problem is one of the easiest problem in graph theory. If the graph is only required to span a subset of nodes, usually called terminals ($T \subseteq V$), then the problem becomes the Steiner tree problem. This problem has been largely studied (see for instance [22], [18]) and is known to be NP-hard in most cases. The Steiner Tree problem has also been considered in the context of a directed graph [23], where a directed tree rooted at some node s and directed towards all the other terminals is called an *arborescence*.

The *Steiner tree packing* problem consists in finding as many disjoint Steiner trees as possible in the same capacitated graph. In the *Fractional Steiner tree packing* problem, a fractional value is to be assigned to each tree so that the weighted sum of all trees using a same edge is less than its capacity. This problem is hence directly related to the problem of finding the maximum throughput in a multicast network when multicast streams can be distributed over several multicast trees. It has been shown to be NP-hard [17].

When trying to maximize the throughput of a single multicast session (Steiner arborescence), one is only concerned with the arc of minimal capacity (which should be maximized). This problem can be seen as a *min-max* (or *max-min*) variant of the Steiner arborescence problem, also known as Bottleneck Steiner Tree problem (BST). Although the "classical" Steiner tree problems are difficult (most of the time NP-hard), their bottleneck counter-parts can usually be solved in polynomial time. Polynomial algorithms with a complexity proportional to the number of edges or arcs have been proposed in [24] and [25] to solve the bottleneck Steiner tree and arborescence problems (in their min-max versions).

Without loss of generality, in this paper, we will describe all models using the maxmin fairness objective function (2). We will also describe all models and algorithms for the directed case.

IV. Models and Algorithms

The network is modeled as a directed capacitated network $G = (V, A, C)$ where $C_a > 0$ is the capacity of arc $a \in A$, $n_V = |V|$, $n_A = |A|$. A set K of streams (or traffic processes, or commodities) must be routed in the network between source and destination nodes. We note $m = |K|$ the number of streams, $s^k \in V$ and $t^k \in V$, the source and destination nodes for the stream $k \in K$. We note $S = \bigcup_{k \in K} \{s^k\}$ the set all source nodes and $T = \bigcup_{k \in K} \{t^k\}$ the set of all destination nodes (*terminals*). In this paper, we assume that all streams share the same source ($|S| = 1$) and denote s the single source.

We denote by $\mathcal{P}^k$ or $\mathcal{P}^{s^k t^k}$ the set of simple paths (without cycles) between s^k and t^k, and by $\mathcal{P} = \bigcup_{k \in K} \mathcal{P}^k$ the set of all paths. Similarly, we denote by $\mathcal{T}(r, U)$ the set of trees rooted at $r \in V$ and spanning $U \subset V \setminus \{r\}$. If the root node or the set of spanned nodes is obvious from the context, they will be omitted in the notation. For any subset of nodes $W \subset V$, we denote $\delta^-_G(W)$ and $\delta^+_G(W)$ the set of arcs leaving and entering W in the graph G. Again, the graph in subscript will be omitted if it is clear from the context.

To model the throughput maximization problems, we will mainly use the following flow variables:

- f^k_a is the total flow of stream $k \in K$ (between s^k and t^k) on arc $a \in A$.
- f_a is the total flow on arc $a \in A$.
- φ^k_p is the total flow of stream $k \in K$ on path $p \in \mathcal{P}^k$ (and simply φ_p for the total flow on path $p \in \mathcal{P}$).
- φ_τ is the total flow on the tree τ.

In this paper, we focus our attention on the particular case where all streams originate from the same source node s. In other words, we do not take into account possible interactions of streams originating from different sources. The multicast models we propose can easily be extended to this multiple source case. However, the multiple source network coding throughput evaluation is much more tricky and we leave this for future research.

We now provide standard (LP=Linear Programming or MIP=Mixed Integer Programming) models for computing the maximum throughput according to the various routing paradigms.

A. Network Coding

As already mentioned, the Network Coding throughput can be computed by applying a maximum flow algorithm for each terminal in T. Denote by $\varphi^*(s,t)$ the value of the maximum flow between s and t:

$$\varphi^*(s,t) = \max \left\{ \sum_{p \in \mathcal{P}^{st}} \varphi_p : \sum_{\substack{p \in \mathcal{P}^{st}: \\ p \ni a}} \varphi_p \leq c_a, \forall a \in A \right\}. \quad (3)$$

Then, the maximum NC throughput is given by:

$$NC : \min_{t \in T} \varphi^*(s,t). \quad (4)$$

Note that this minmax problem can then be easily linearized into a single LP (in a fashion similar as in [21]). We provide the model here merely to show some similarities with the other models:

$$NC_1 \begin{cases} \max \lambda, & (5) \\ \sum_{p \in \mathcal{P}^{st}} \varphi_p \geq \lambda, & \forall t \in T, \quad (6) \\ f_a \geq \sum_{\substack{p \in \mathcal{P}^{st}: \\ p \ni a}} \varphi_p, & \forall a \in A, t \in T, \quad (7) \\ f_a \leq c_a, & \forall a \in A, \quad (8) \\ \varphi_p, f_a \geq 0, & \forall a \in A, p \in \mathcal{P}^{st}. \quad (9) \end{cases}$$

Constraints (6) ensure that the relative amount of flow carried for each stream is at least λ. The right-hand-side in constraints (7) is the flow carried on arc a for the stream between s and t. So the left-hand-side is larger than the maximum of all these flows and represents the equivalent coded flow on arc a. Constraints (8) are the capacity constraints. Note that the variables f_a are not really necessary in this model. We introduce them here to indicate how the resulting NC flow on each arc can be obtained. Also note that the problem could also be formulated using arc-flow variables f^t_a (which are then equal to the right-hand side of constraints (7)) and standard flow conservation constraints.

B. Multicast routing on multiple trees

Here we assume that several multicast trees can be used to forward the traffic from the source s to all destinations T. There are several ways to model this problem. We provide here a simple model based on the assumption that we are able to generate efficiently trees spanning T (the so-called *Steiner trees*):

$$MC_1 \begin{cases} \max \sum_{\tau \in \mathcal{T}} \varphi_\tau, & (10) \\ \sum_{\substack{\tau \in \mathcal{T} \\ \tau \ni a}} \varphi_\tau \leq c_a, & \forall a \in A, \quad (11) \\ \varphi_\tau \geq 0, & \forall \tau \in \mathcal{T}. \quad (12) \end{cases}$$

If we denote by w_a the dual variable associated with the capacity constraints (11), then the reduced cost of tree τ is:

$$\overline{r}_\tau = 1 - \sum_{a \in \tau} w_a. \quad (13)$$

The pricing problem then consists in finding a minimum weight tree (in the graph weighted by the dual variables w_a) rooted at s and spanning the subset of terminals $T \subset V$. When $\{s\} \cup T \neq V$, this problem turns out to be a Steiner tree problem which can be sumarized as follows:

$$ST(w) : \min_{\tau \in \mathcal{T}(T)} \operatorname{sum}_{a \in \tau} w_a. \quad (14)$$

Although the problem is NP-hard, there exists efficient approaches that allow to solve exactly instances, at least of

limited size (up to a hundred nodes). For instance, one can solve the following Mixed-Integer sub-problem:

$$ST(w)\begin{cases}\min \sum_{a\in A} w_a x_a, & (15)\\ x_a \geq r_a^k, & \forall a,k,(16)\\ \sum_{a\in\delta^-(v)} r_a^k - \sum_{a\in\delta^+(v)} r_a^k = b_v^k(1), & \forall v,k,(17)\\ r_a^k \geq 0, x_a \in \{0,1\} & \forall a,k,(18)\end{cases}$$

where the constants b_v^k are defined as:

$$b_v^k(x) = \begin{cases} -x & \text{if } v = s\\ x & \text{if } v = t^k\\ 0 & \text{otherwise}\end{cases} \quad (19)$$

For each stream k, a set of flow variables r_a^k is used to connect the source s with the destination t^k (using standard flow conservation constraints (17)). A binary variable x_a is used to count each arc used by some streams. Since the objective minimizes the cost of arcs, the solution of this problem will hence be a tree spanning T. As already mentioned, the problem (MC_1) is nothing else than a *Fractional Steiner Tree Packing* problem.

Since the number of trees used in a multicast solution is a main concern (because it impacts greatly whether solutions can be implemented in practice), we also considered a second phase algorithm where, the optimal throughput value being fixed, we minimize the number of trees used (which might be smaller than the number of trees used in the optimal solution).

C. Multicast routing on a limited number of trees

Limiting the number of trees in the multicast model adds a significant difficulty to the model. Indeed, it is necessary to count the trees used to carry some traffic, so additional binary variables must be introduced into the models:

$$MC_1^\ell\begin{cases}\max \sum_{\tau\in\mathcal{T}} \varphi_\tau, & (20)\\ \sum_{\substack{\tau\in\mathcal{T}\\ t\ni a}} \varphi_\tau \leq C_a, & \forall a\in A, \quad (21)\\ \varphi_\tau \leq C x_\tau, & \forall \tau\in\mathcal{T}, \quad (22)\\ \sum_{\tau\in\mathcal{T}} x_\tau \leq \ell, & (23)\\ \varphi_\tau \geq 0, x_\tau \in \{0,1\} & \forall \tau\in\mathcal{T}. \quad (24)\end{cases}$$

Here, the constraint (23) limits the number of trees used in a solution to a maximum of ℓ. The constant C can be set to the maximum of all capacities. Constraints (22) are used to set to 0 the flows on trees when the associated binary variable is equal to 0.

Since it is intractable in practice to consider explicitly all possible trees of $\mathcal{T}$, to solve (MC_1^ℓ), one must again consider column generation phases where a pricing sub-problem is solved to find candidate trees to add into the master problem (as in IV-B). However, since the master problem is a MIP, its resolution involves branching phases and the pricing subproblem must be solved in each node of the branch-and-bound tree (branch-and-price). Moreover, the pricing problem has to be adapted to fit each local version of the master problem.

One way to avoid this tedious resolution process consists in relying on simple heuristics where a limited set of candidate trees $\hat{\mathcal{T}} \subset \mathcal{T}$ is generated in a first phase (for instance by considering various perturbed versions of the initial master problem) and then the MIP is solved once with this unique set $\hat{\mathcal{T}}$:

Algorithm 1 : Simple heuristic for MC_1^ℓ

Input: directed capacitated graph $G = (V, A, C)$, set of single-source streams $\{s, t^k, d^k\}_{k\in K}$;

Output: a lower-bound on the multicast maximum throughput using less than ℓ trees;

1. Solve Problem MC_1;
2. If $|\mathcal{T}^{MC}| \leq \ell$, then the continuous solution of MC is also optimal for MC_1^ℓ: stop the algorithm;
3. Otherwise, let $\hat{\mathcal{T}} \leftarrow \mathcal{T}^{MC}$;
4. Solve problem MC_1^ℓ defined over the restricted set of trees $\hat{\mathcal{T}}$;

Note that, when ℓ, the number of trees, is small, we have also considered an alternate exact formulations for (MC_1^ℓ).

V. Numerical experiments

The previous section provides guidelines to solve the considered maximum throughput problems. We have used a commercial tool (XPress Optimizer Version 21.01.00) to solve LP (Linear Programming) and MIP (mixed Integer Programming) problems. We have conducted several computational experiments on directed and on bi-directed euclidean instances. These instances were randomly generated using algorithm 2. Series of 100, 1000 or even 10,000 random instances have been generated for several sets (n, m, k_s, k_t). Note that for bidirected instances, m represents the number of links, so that the number of arcs is $2m$. For directed instances, for each edge $\{i, j\}$ to generate, we decide randomly to introduce the arc (i, j), the arc (j, i) or both at a time. We have applied our maximum throughput algorithms are on all these instances and average results are reported.

Algorithm 2 : Generate Directed/Bidirected Instance

Input: number of nodes n, number of arcs m, number of sources k_s, number of terminals k_t, a geographical box $\Omega = [a_1, b_1] \times [a_2, b_2]$, a capacity interval $[C_{min}, C_{max}]$;

Output: a connected capacitated undirected graph $G = (V, E, C)$ within Ω;

1. Generate uniformly n nodes in $\Omega \rightarrow V$;
2. Compute a minimum distance spanning arborescence/tree in the complete graph K_n;
3. Randomly add $m - n + 1$ arcs/edges with a probability inversely proportional to the distance between the end-nodes;
4. Randomly generate arc/edge capacities in $[C_{min}, C_{max}]$;

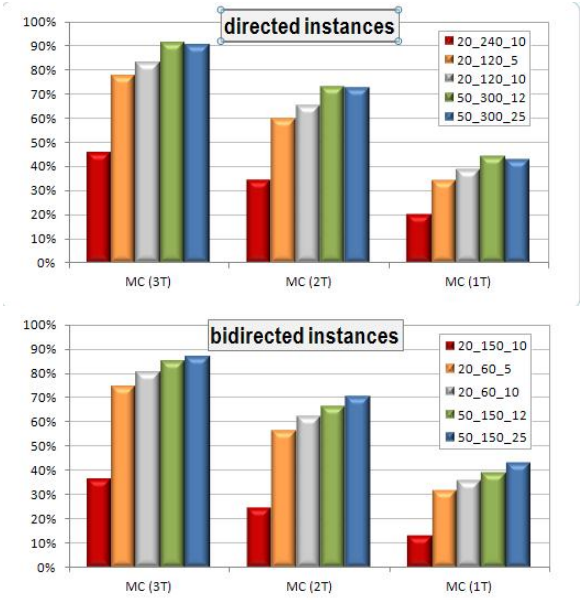

Fig. 3. Relative throughput achieved for MC with 1, 2 and 3 trees over different sets of instances (given by n, m and kt, all with a single source: $ks = 1$). The values provided are the average over 100 random instances of each type. The maximum (100 %) corresponds to the NC and unrestricted MC throughputs.

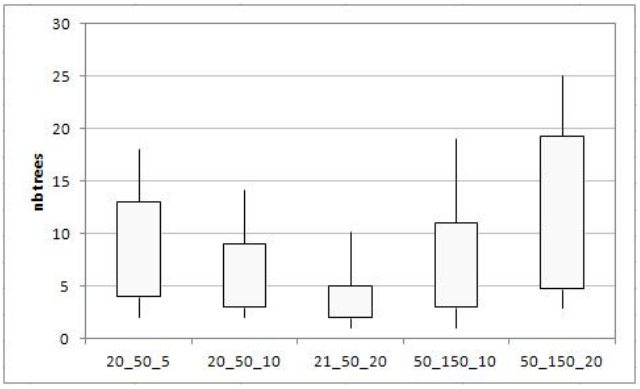

Fig. 4. For each set (indicated on the x-axis), 100 random instances are generated and the minimal number of multicast trees required to obtain the optimal throughput is computed. The resulting values for 50% of the instances fall within the boxes (and for 90 % within the intervals).

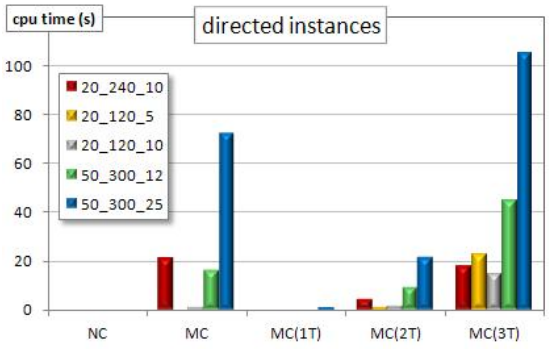

Fig. 5. Average computing times (in seconds) for solving the various maximum throughput problems.

The first surprising result is that, over all the generated instances, directed and bidirected, we could not find a single one where network coding (NC) had a larger maximum throughput than multicast (MC). However, when we considered multicast with a limited number of trees (MC-ℓ), the situation was quite different. Figure 3 shows the relative throughput values (100% means NC and MC throughput) achieved when restricting multicast to use only 1, 2 or 3 trees. We see that the throughput value decrease as the number of trees decreases. The general trends are very similar for the directed and bidirected instances. The reduction of throughput is much more important when the instances are denser (about $6n$ edges for the first instance set and $3n$ edges for the other). This is due to the fact that a limited number of trees cannot exploit the full potential offered by the network whereas Network Coding does. In traditional telecommunication networks, the average degree is usually rather low (say between 3 and 5). The observations made on the four last series of instances show that there is still a significant throughput reduction (from 13 to 25 %) when using up to 3 multicast trees, when compared to a network coding solution. Network coding should hence be considered by network administrators as a very attractive alternative solution to standard routing solutions.

Figure 4 gives some insight on the number of trees required in a multicast setting to reach the same throughput as if NC was used. The resulting numbers fall within the boxes for 50 % of the generated instances (and within the intervals for 90 % of the instances). We can observe that the variance in the optimal number of trees is quite high, some instances requiring relatively few trees whereas others require a large number of trees. The ranges and values decrease when the number of terminal increases. This can be related to the fact that, thanks to Edmonds arborescence packing theorem [26], the theoretical gain of Network Coding throughput vanishes when ALL nodes are terminal. If five trees would be considered as a reasonable upper-limit for operators to handle, then, in most case, the NC trhoughput would not been achieved with multicast. On the other hand, ten trees would often suffice for small networks.

In Figure 5, we provide the average, over all generated instances of a given series, of the computing times for obtaining the optimal throughput values using our models. We clearly see that multicast throughput computations generally takes a few minutes whereas network coding throughput computation is instantaneous. Computations for a single multicast tree are also very fast, but for values larger than 2, the problem becomes increasingly long to solve. We have performed many other computational experiments, including some with multiple source instances, and the main observations remain generally the same.

VI. Considerations on some small instances

The main outcome of our first numerical experiments is that Network Coding (NC) never seems to outperform throughput

Set	n	m	ks	kt	λ_{NC}	λ_{MC}	# trees
Instances n_rand_rand							
AVG	7	14.35	1	4.03	2.95	2.95	4.83
MIN	7	8	1	2	1	1	1
MAX	7	21	1	6	6	6	30
AVG	8	18.24	1	4.52	3.25	3.25	6.15
MIN	8	9	1	2	1	1	1
MAX	8	28	1	7	7	7	42
AVG	9	22.71	1	5	3.54	3.54	7.3
MIN	9	10	1	2	1	1	1
MAX	9	36	1	8	8	8	56
AVG	10	27.81	1	5.43	3.9	3.9	9.08
MIN	10	11	1	2	1	1	1
MAX	10	45	1	9	9	9	72

TABLE I
(AVERAGE OVER 1000 INSTANCES) RESULTS FOR THE SINGLE-SOURCE CASE WITH UNIFORM CAPACITY $C_a = 1, \forall a$.

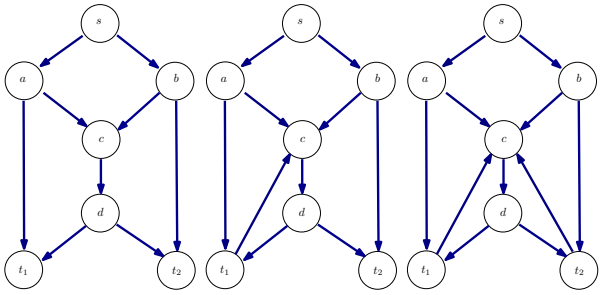

Fig. 6. The only 3 graphs with a non-zero gap (of 0.5) between NC and MC throughput (in the case of uniform capacity $C_a = 1, \forall a$).

allowed by Multicast solutions (MC). Focusing more on small size instances, the results in Table I show that all uniform instances (all capacities are equel to 1) generated with 7 to 10 nodes have the same maximum throughput with NC and MC. In particular, this means that our generator was not able to reproduce the "classical" butterfly network. To get some insight on this surprising phenomena, we decided to perform an exhaustive search for the restricted case of graph with 7 nodes, 1 source, 2 terminals and all capacities equal to 1. To limit somewhat the search, we have only considered instances with at least two outgoing arcs from the source and with at least two incoming arcs into the terminals. Among the 950,951 possible graphs (removing 18,016 disconnected cases), only 96 showed a non-zero gap between NC and MC. In fact, these 96 cases can be essentially reduced to the 3 graphs displayed in Figure 6, all other cases being symmetrically equivalent to those 3. Observe that the first one is the classical butterfly graph, whereas the two other are small variants. If we consider a uniform distribution among all graphs, then the probability to pick a graph with a non-zero gap is about 0.01%.

One step further in this line of research consists, instead of enumerating all possible instances of a certain type, in considering the problem of finding the maximum gap as an optimization problem. For that purpose, we define binary design variables $x_a \in \{0, 1\}$ which will be set to 1 for the arcs that are kept in the final topology (assuming we start from a potential fully meshed topology, i.e., a complete graph). We will also assume that the number of nodes n, the number of streams m are given, as well as candidate capacities C_a, one for each arc. Here, we will mainly consider uniform capacity models ($C_a = 1, \forall a \in A$) and single source models. Hence, in the instances, kt will specify the number of terminals. So, for instance, the setting $(n = 10, kt = 5)$ concerns all instances with 10 nodes, one source node and 5 terminals. The optimization problem then consists in finding the subgraph of K_n with kt terminals, where the gap between NC and MC throughput is maximal. So basically, the problem could be summarized as:

$$\Delta^*(NC, MC) = \max_{x_a \in \{0,1\}} \left(\lambda^*_{NC} - \lambda^*_{MC}\right), \qquad (25)$$

where λ^*_{NC} and λ^*_{MC} are the maximum throughput achieved by NC and MC, and are thus, the results from their own maximization problem. For the NC throughput maximization problem, we are going to rely on arc-flow variables (instead of flow-path variables as used up to now): $\lambda^*_{NC}(\mathbf{x}) =$

$$\lambda^*_{NC}(\mathbf{x}) = \begin{cases} \max \lambda, & (26) \\ \sum_{a \in \delta^-(v)} r^k_a - \sum_{a \in \delta^+(v)} r^k_a = b^k_v(\lambda), & \forall v, k, (27) \\ r^k_a \le c_a x_a & \forall a, k, (28) \\ r^k_a \ge 0, & \forall a, k. (29) \end{cases}$$

Since we need to substract MC maximum throughput in our global problem, we choose to express λ^*_{MC} in a LP-dual form:

$$\lambda^*_{MC}(\mathbf{x}) = \begin{cases} \min \sum_{a \in A} c_a x_a v_a, & (30) \\ \sum_{k \in K} (\pi^k_s - \pi^k_{t^k}) = 1, & (31) \\ u^k_{ij} \ge \pi^k_i - \pi^k_j, & \forall i, j, k, \quad (32) \\ v_a \ge \sum_{k \in K} u^k_a, & \forall a, k, \quad (33) \\ \pi^k_i, u^k_a, v_a \ge 0, & \forall a, i, k. \quad (34) \end{cases}$$

This is the standard dual formulation of an equivalent arc-flow formulation of MC_1 (eq. (10)-(12)). Note that, when we consider both problems jointly in (25), the objective of the MC throughput optimization subproblem becomes non-linear (product of variables x_a and v_a). Using standard linearization techniques, we can obtain a single MIP formulation for the joint problem (25). As it is often the case in such linearized problems, the continuous relaxation bound is very weak. To further tighten the model, we have also used series of symmetry breaking constraints, in order to avoid to consider multiple similar solutions.

Plugging this model in the Xpress solver, we were able to solve a few small size instances (see Table II). It is interesting to note that, for uniform instances with 7 and 8 nodes, the only cases with a non-zero gap between NC and MC maximum throughput are 7 nodes with 2 destinations, 8 nodes with 2 or 3 destinations. The first case corresponds to the classical "butterfly network". Two graphs for the 8 nodes, 2 or 3

n	k	c_a	$\Delta^*(NC, MC)$	λ^*_{NC}	λ^*_{MC}	m	cpu (sec)
7	2	1	0.5	2	1.5	9	0.8
7	3	1	0	-	-	-	0.2
7	4	1	0	-	-	-	0.1
7	5	1	0	-	-	-	0.1
7	6	1	0	-	-	-	0.1
8	2	1	0.5	3	2.5	13	380
8	3	1	0.5	2	1.5	11	12
8	4	1	0	-	-	-	0,5
8	5	1	0	-	-	-	0,1
8	6	1	0	-	-	-	0,1
8	7	1	0	-	-	-	0,1

TABLE II
RESULTS FOR COMPUTATION OF MAXIMUM THROUGHPUT BETWEEN NC AND MC (UNIFORM INSTANCES).

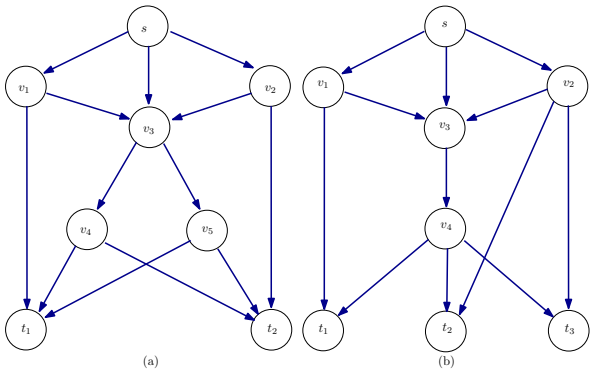

Fig. 7. Two 8 nodes graphs whit a 0.5 gap between uniform NC and MC throughput maximum flows.

terminals case are displayed in Figure 7. It is easy to check that, in these to graphs, the gap is indeed 0.5.

Note that we have used here an experimental way of establishing the following result:

Lemma 1: consider directed networks with n nodes ($n = 7$ or 8), unit capacities $c_a = 1, \forall a \in A$), a single source and k terminals (different from the source node). If $n = 7$ and $k \geq 3$ or $n = 8$ and $k \geq 4$, then network coding does not improve the throughput with respect to standard multicast.

VII. CONCLUSION

In this paper, we have used elaborate resolution algorithms, mainly based on LP and MIP models, to solve large sets of throughput maximization problems, using either Multicast routing (MC) alone, or combined with Network Coding (NC) techniques. Our main finding is that situations where network coding improves the throughput almost never occur, and is only obtained in very specific instances, and this observation holds for directed and bi-directed settings. However, when we consider routing scenarios where the number of multicast trees is limited, Network Coding cleary allows a very significant improvement in throughtput, let alone all other advantages of Network Coding that have already been largely advertise in the literature.

REFERENCES

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

[2] L. Ford and D. Fulkerson, "Maximum flow through a network, research memorandum rm-1400," The RAND Corporation, Santa Monica, California, Tech. Rep., 1954.

[3] ——, "Maximum flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[4] T. Ho and D. S. Lun, *Network Coding an introduction*. New York, USA: Cambridge University Press, 2008.

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

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

[7] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[8] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "Toward a random operation of networks," *IEEE Transactions on Information Theory*, vol. 2004, pp. 1–8, 2004.

[9] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413 –4430, oct. 2006.

[10] H. Wang, J. Liang, and C.-C. J. Kuo, "Overview of robust video streaming with network coding," *Journal of Visual Communication and Image Representation*, 2010.

[11] M. Montpetit and M. Medard, "Video-centric network coding strategies for 4g wireless networks: An overview," in *Consumer Com. and Net. Conf. (CCNC), 2010 7th IEEE*, jan. 2010, pp. 1–5.

[12] M. Kim, M. Médard, and J. Barros, "Modeling network coded tcp throughput: A simple model and its validation," in *Valuetools*, 2011.

[13] L. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: A tutorial," *IEEE Networks*, vol. 14, pp. 90–102, 2000.

[14] Z. Li, B. Li, and L. C. Lau, "A constant bound on throughput improvement of multicast network coding in undirected networks," *Information Theory, IEEE Transactions on*, vol. 55, no. 3, pp. 1016 –1026, march 2009.

[15] A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency*. Berlin Heidelberg: Springer-Verlag, 2003.

[16] N. Garg, R. Khandekar, K. Kunal, and V. Pandit, "Bandwidth maximization in multicasting," in *Proceedings of European Symposium on Algorithms (ESA)*, 2003, pp. 242–253.

[17] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing steiner trees," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '03, 2003, pp. 266–274.

[18] M. Goemans and Y. Myung, "A catalog of steiner tree formulations," *Networks*, vol. 23, pp. 19–28, 1993.

[19] T. Koch and A. Martin, "Solving steiner tree problems in graphs to optimality," *Networks*, vol. 32, pp. 207–232, 1998.

[20] Z. Li, B. Li, D. Jiang, and L. C. Lau, "On achieving optimal throughput with network coding," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, march 2005, pp. 2184 – 2194.

[21] D. S. Lun, M. Médard, T. Ho, and R. Koetter, "Network coding with a cost criterion," in *Proc. 2004 International Symposium on Information Theory and its Applications (ISITA 2004*, 2004, pp. 1232–1237.

[22] J. Beasley, "An algorithm for the steiner problem in graphs," *Networks*, vol. 14, pp. 147–159, 1984.

[23] L. Z. Samir and S. Khuller, "On directed steiner trees," in *13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002, pp. 59–63.

[24] C. Duin and A. Volgenant, "The partial sum criterion for steiner trees in graphs and shortest paths," *European Journal of Operations Research*, vol. 97, pp. 172–182, 1997.

[25] L. Georgiadis, "Bottleneck multicast trees in linear time," *IEEE Communications Letters*, vol. 7, no. 11, pp. 564–566, 2003.

[26] J. Edmonds, "Edge-disjoint branchings," *Combinatorial Algorithms, Algorithmics Press, New York*, pp. 91–96, 1972.