REPORTS
IN
INFORMATICS

ISSN 0333-3590

Heuristic Methods for Flow Graph Selection in Integral Network Coding

Mohammad Ravanbakhsh and Dag Haugland

REPORT NO 391 **October 2009**

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL
http://www.ii.uib.no/publikasjoner/texrap/pdf/2009-391.pdf

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Heuristic Methods for Flow Graph Selection in Integral Network Coding

Mohammad Ravanbakhsh*and Dag Haugland†
Department of Informatics
University of Bergen
N-5020 Bergen
Norway

October 26, 2009

Abstract

The search for optimal multicast subgraphs for network coding is considered. We assume unit link capacities and binary flow rates. In the first version of the problem, there is no constrained on the acyclicity of the subgraphs, whereas such constraints are imposed in the second version. These problems are known to be NP-hard. We provide heuristics to deal with both versions of the problem. The heuristics are based on well known optimization algorithms and they are therefore easy to implement.

1 Introduction

Multicast in a communication networks is a mode of operation where, a group of source nodes transmits to a group of sink nodes, and where for each source, the information transmitted is identical for all sinks. The capacity of a multicast session is the largest number of bits that hence can be transmitted in a given time unit. Traditionally, the transmission is based on simple forwarding of entering packets, implying that the multicast capacity is given by the solution to a maximum flow problem, or equivalently, a minimum cut problem. The problems in question are formulated on a graph representing the communication network, where link capacities reflect the maximum transmission rate between pairs of connected nodes. As pointed out by Ahlswede et al. [2], the capacity can be increased substantially if the routers are enabled to combine several entering transmission packets and re-encode before forwarding the packets. This process, known as *network coding* is particularly useful in applications where the flow capacity is small in comparison to the number of users, such as fixed or mobile wireless networks.

To accommodate best possible utilization of network resources, any multicast session should be established by first identifying a subset of resources to be assigned to the session. Thus, the first step of a network coding scenario, which can be referred to as the *flow graph selection* step, is to select the network links to carry the flow from source nodes to sink nodes. The selected links induce a subgraph of the graph representing the communication network, and is in this work referred to as the *flow graph*. In the second step, a valid

*Email: mohammad.ravanbakhsh@ii.uib.no, WWW: http://www.ii.uib.no/~mohrav
†Email: Dag.Haugland@ii.uib.no, WWW: http://www.ii.uib.no/~dag

network encoding corresponding to the flow graph is provided. Either randomly network encoded packets[3], or, more efficiently when the network is stable, a fixed network encoding scheme can be designed [4, 5, 6] for the multicast session on the selected flow graph.

In this work, we focus on the challenge of accomplishing the first step in the best possible way. In general, for a given network and a given multicast request, there exist a set of flow graphs that serve the needs of the multicast session. The problem is therefore defined in terms of a cost minimization model, where the cost is a sum of costs associated with each link included in the flow graph. Link costs typically reflect latency, power consumption or other metrics, or possibly a combination of these, associated with pairs of connected nodes. A corresponding optimization algorithm will thus attempt to find a "best" flow graph as defined by the chosen criterion.

As demonstrated in [7], the flow graph selection problem can be recognized as the one of computing a set of paths from a unique source to each terminal, such that the paths corresponding to any sink are link disjoint, and such that the total link cost is minimized. By a reduction from the minimum Steiner tree problem, it is seen that this problem is NP-hard [7]. To avoid excessive computations in the first step, the focus in the current work is therefore on fast methods producing near-optimal solutions.

Two different variants of the flow graph selection problem are studied in this work. First, we consider no flow graph constraints but the link disjoint property of paths leading to a common sink. In the second variant, however, we also impose the requirement that the flow graph has no directed cycles. In other words, the flow graph must be a *Directed Acyclic Graph* (DAG), which is required in many network coding algorithms, such as e.g. LIF [4].

1.1 Contributions of this paper

In this paper, we propose heuristics for the flow graph selection problem in network coding. We focus on graphs with unit rate capacity links, since problems with integer capacity links easily can be transformed to parallel links with unit rate capacity. For the first variant of the problem, we propose a method that guarantees to extract a flow graph with integer flow rates (as opposed to real numbered ones), given that such flow graphs exist. We claim and justify that our approach is simpler and faster than the method suggested in [7]. For the acyclic, and hence more challenging, version of the problem, we also propose a fast heuristic. However, it cannot be claimed that this method is successful in finding a feasible solution (an acyclic flow graph) to all instances for which such solutions exist.

The structure of this paper is as follows. In Section 2 we describe the network model in more detail, and we introduce the notation. We formulate the proposed heuristics in Section 3. Finally, we present the conclusions in Section 4.

2 Notation and network model

A communication network can be described by a directed graph $G = (N, E)$, where N is the set of nodes and E is the set of links. A *node* represents a router, station, or host computer, and the presence of a link $(i, j) \in E$ expresses that node i can communicate directly with node j. Each node will have a unique *location* in the plane. In our application, the set of nodes N can be partitioned into three disjoint sets, namely S the set of *sources*, T the set of *sinks* (or *terminals*), and R the set of *intermediate nodes*. For each link $(i, j) \in E$, we assume there is a fixed cost denoted a_{ij} of using the link in a multicast session. The links will be assumed to have *unit capacity*, and below we argue why this is a reasonable assumption.

If the input rate of the multicast session is n_S, where n_S is some positive integer, we consider a network with $n_S = |S|$ source nodes, each producing a unit rate data stream. This implies no serious restriction since a source with integer rate n_S can be modeled as n_S co-located sources, each one of rate 1. When applying the flow algorithms and the network coding algorithms, we can transform the problem into one with a single source by introducing a single *dummy source* node with only n_S outgoing links connected to source nodes. The purpose of the multicast session that we consider is to transmit all information from the dummy source to each of the $|T|$ sinks.

The following two subsections describe in more detail the network that we consider.

2.1 Flows

Definition 1. *A* flow[1] *of a given sink $t \in T$ is a set of link disjoint paths to t, one from each source in S.*

Observe that flows corresponding to different sinks may share links.

Definition 2. *A* flow graph *in G is a subgraph of G where the union of flows, one for each sink in T, constitutes the link set.*

The flow graph of G is not necessarily unique. The main theorem in network coding implies that a network represented by G supports a multicast session as long as there exists at least one flow graph in G.

The centralized network encoding algorithms described in the literature are typically applied to a flow graph in G, sometimes with conditions on acyclicity. In [4], network coding is performed on a link by link basis by dynamic programming, provided the flow graph is directed and acyclic.

In this work, we study two versions of the problem of finding a flow graph in G with minimum total link cost. In the *cyclic* version, we allow the flow graph to be cyclic, whereas absence of directed cycles is an imposed constraint in the *acyclic* version. Define $z \in \{0,1\}^E$ such that for all $(i,j) \in E$ $z_{ij} = 1$ if and only if link (i,j) is included in G, and for all $t \in T$, define $x^t \in \Re^E$ as the vector of link flows destined for sink t. Then the cyclic version of the problem can be stated as

$$\begin{aligned}
&\text{minimize} && \textstyle\sum_{(i,j)\in E} a_{ij} z_{ij} && && (1)\\
&\text{subject to:} && && && \\
& && 0 \le z_{ij} \leqslant 1 && \forall (i,j) \in E && (2)\\
& && 0 \leqslant x^t_{ij} \leqslant z_{ij} && \forall (i,j) \in E,\ \forall t \in T && (3)\\
& && \sum_{(i,j)\in\Gamma_O(i)} x^t_{ij} - \sum_{(j,i)\in\Gamma_I(i)} x^t_{ji} = \Delta^t_i && \forall i \in N,\ \forall t \in T && (4)\\
& && z_{ij} \text{ integer} && \forall (i,j) \in E. && (5)
\end{aligned}$$

where s is the dummy source node and, $\Gamma_O(i)$ and $\Gamma_I(i)$, are the sets of outgoing and incoming links in node i, respectively, and

$$\Delta^t_i = \begin{cases} n_S, & \text{if } i = s \\ -n_S, & \text{if } i = t \\ 0, & \text{otherwise.} \end{cases} \qquad (6)$$

[1] This definition is motivated by unit capacity links and integer flow values.

In the acyclic version, the additional constraints can be formulated as:

$$u_{j,k-1} \geq u_{jk} \geq u_{i,k-1} + z_{ij} - 1 \quad \forall (i,j) \in E,\ \forall k = 1, \ldots, |N| \tag{7}$$

$$u_{i|N|} = 0 \quad \forall i \in N, \tag{8}$$

$$u_{s0} = 1 \tag{9}$$

$$u_{ik} \in \{0,1\} \quad \forall i \in N,\ \forall k = 0, \ldots, |N|. \tag{10}$$

where for all $i \in N, k = 0, \ldots, |N|$,

$$u_{ik} = \begin{cases} 1, & z \text{ defines some flow path from } s \text{ to } i \text{ of at least } k \text{ links} \\ 0, & \text{otherwise} \end{cases}$$

The first inequality in constraint (7) states the obvious fact that if $j \in N$ lies on a path of length k or more, than it also lies on a path of length at least $k-1$. The second inequality says that if $i \in N$ is on a path of length at least $k-1$, and $(i,j) \in E$ is in the flow graph, then j is on a path of length at least k. Combined with (8)-(9), these constraints ensure absence of directed cycles.

3 Flow graph construction methods

In many deterministic algorithms used for network encoding (e.g. Jaggi et al. [4]), it has been assumed that for an integral rate (n_S) network coding, there are n_S link-disjoint paths from the (dummy) source node to each sink (recall $n_S = |S|$, where $|S|$ is the number of unit rate sources). In [9], linear programming (LP) is introduced to compute the optimal subgraph for a multicast session. The LP-method in [9] produces solutions in the field of *real numbers*. In this format, the subgraph which carries the flow for a specific sink is not necessarily a set of link-disjoint paths, and in such cases, it is not a flow. On the other hand, restricting the LP to have integer solutions raises a problem that is known to be NP-hard [7]. In [7], approximation algorithms are introduced to produce integral solutions. The method used in [7] is based on iteratively solving the minimum-cost flow problem as defined in [9] by modifying the link costs.

For both versions of the problem defined in the previous section, we now suggest methods to construct flow graphs in G. The first method, described in Section 3.1, finds a feasible solution to the cyclic version based on considerations of the solution to the LP (1)-(4). This method does not guarantee an optimum integer solution, but as long as the instance is feasible, the method guarantees that a feasible solution is found. In practice, the solution can also be close to optimum. In Section 3.2, we modify the well-known Ford-Fulkerson algorithm [1] in order to extract flow graphs for the acyclic problem.

3.1 LP-based methods for the cyclic version

In this section, we suggest a method for constructing flows that may or may not be cyclic. In contrast to [7], our approach needs to solve the LP problem (1)-(4) only once, after which the flow graph is extracted by solving a minimum cost flow problem for each sink. Thus, the running time of our approach is roughly that of solving (1)-(4) plus the time it takes to solve $|T|$ much smaller LP-problems. We have not compared the performance of our approach with [7], but in many instances, the solution it produces is close to the relaxed optimal solution. Also, there are two main reasons for introducing a new method based on LP instead of using [7]. First, in our algorithm we need to solve (1)-(4) once, whereas each iteration of [7] requires to solve a modified version of (1)-(4). Hence, our approach is faster. Second, for the class of problems where links have unit rate capacity, we show in

Theorem 1 that in any instance, feasible integer solutions based on the solution to (1)-(4). Our flow graph selection method consists of the following two steps:

3.1.1 First step of the flow selection method

Solving the optimization problem defined in [9] yields a possibly fractional solution, of which the non-zero variables define a subgraph. We later use this subgraph in the second step to generate flows by applying a method based on integer programming, the details of which will be explained below.

Assume (1)-(4) is feasible, and denote its optimal solution by $(\hat{x}, \hat{z})$.
Define $E_t = \left\{(i,j) \in E : \hat{x}_{ij}^t > 0\right\}$. That is, E_t consists of the links that in the optimal solution to (1)-(4) carry flow from the dummy source to sink t. Since integrality constraints are relaxed in this problem, the paths defined by positive $\hat{x}_{ij}$-values are not necessarily link-disjoint, and E_t is not necessarily a flow according to Definition 1.

Theorem 1. *If (1)-(4) is feasible, then for all $t \in T$, $G_t = (N, E_t)$ contains n_S link-disjoint paths from s to t.*

Proof. We prove that if all links in G_t have unit capacity, n_S units of flow can be sent from s to t in G_t. This condition is a equivalent to the existence of n_S link-disjoint paths from s to t in G_t.

First, consider the problem of maximizing the flow from s to sink $t \in T$ in G when the links $(i,j) \in E$ have capacities $\hat{z}_{ij}$. We have that $\hat{x}^t$ is a feasible solution to this problem since $(\hat{x}, \hat{z})$ is feasible in (1)-(4). By (4) and (6), we also have that the corresponding total flow is n_S. Because $\hat{z}_{ij} \leq 1$ for all links (i,j), we have that $\hat{x}^t$ remains feasible in the max flow problem if the capacities are increased to 1 for all (i,j) for which $\hat{z}_{ij} > 0$. Hence, we can send n_S units of flow from s to t in G_t given that all its links have unit capacity. □

In the favorable case where $\hat{x}$ happens to be integral, we have solved the problem. Otherwise, we go on with a second step which for all $t \in T$ computes a flow in G_t.

3.1.2 Second step of the flow selection method

In the second step, we go through the sinks sequentially, and for each $t \in T$, we compute a flow $B_t \subseteq E_t$. By Theorem 1, we know that such a flow exists. Assume that a subset $T' \subseteq T$ hence has been processed. For all $(i,j) \in E_t$, we define the decision variables $y_{ij}^t = \begin{cases} 1, & (i,j) \in B_t \\ 0, & \text{otherwise,} \end{cases}$ and let $a_{ij}^t = \begin{cases} \frac{1}{\hat{x}_{ij}^t}, & (i,j) \notin B_{t'}\ \forall t' \in T' \\ 0, & \text{otherwise} \end{cases}$ represent the cost of including (i,j) in the flow corresponding to $t \in T \setminus T'$. Thus, the following optimization problem is solved in order to find B_t:

$$\text{minimize} \sum_{(i,j) \in E_t} a_{ij}^t y_{ij}^t \tag{11}$$

subject to:

$$\sum_{(i,j) \in \Gamma_O(i) \cap E_t} y_{ij}^t - \sum_{(j,i) \in \Gamma_I(i) \cap E_t} y_{ji}^t = \Delta_i^t \ \forall i \in N \tag{12}$$

$$y_{ij}^t \in \{0,1\}\ \forall (i,j) \in E_t \tag{13}$$

The motivation for applying the costs a_{ij}^t is that if (i,j) has already been included in a flow $B_{t'}$, the additional cost of including it in B_t is zero. Further, putting the cost to $\frac{1}{\hat{x}_{ij}^t}$, if (i,j) is not yet included in any flow, encourages reuse of links $(i,j) \in E_t$ for which x_{ij}^t, and thereby z_{ij}, have large optimal values in (1)-(4). Thus the resulting link-disjoint paths are likely to constitute a subgraph close to the optimal solution to (1)-(5).

Problem (11)-(13) is recognized as a minimum cost flow problem, and is thus solved quickly.

3.2 Flow Construction Algorithms with Control of Cyclicity

In the approach presented in the previous section, we have no control on the existence of cycles in the flows produced. In order to control the occurrence of cycles, we give in this section a flow-augmenting approach based on the Ford-Fulkerson algorithm.

Algorithm 1 works on the *residual graph* [8] of G at each step. Initially, all links $(i,j) \in E$ are assigned the fixed cost a_{ij}, but once we have that $(i,j) \in B_{t'}$ for some $t' \in T'$, we define the cost of link (i,j) to be zero. The purpose of this update is to stimulate link sharing among sinks.

Since we assume unit capacity links and integer flow values, the residual graph is obtained from G by *inverting* the direction of links with current flow equal to one and *keeping* the links with flow equal to zero. The sign of the cost of any inverted link is correspondingly reversed. An augmenting path [8] is simply a path from the dummy source to the sink in that residual graph. We refer to [8] for a more detailed and general explanation of the Ford-Fulkerson algorithm and the significance of augmenting paths.

We find the augmenting paths by using Dijkstra's algorithm, which locates paths of minimum cost. In order to avoid cycles, Dijkstra's algorithm is modified to check whether the addition of a link makes the flow graph cyclic, in which case the link is ignored.

Unlike the approach suggested for the cyclic version, Algorithm 1 does not guarantee to output a feasible solution even when such solutions exist.

Obviously, the flow-augmenting approach can also be applied without modification of Dijkstra's algorithm in order to find a flow graph for the cyclic version of the problem.

Algorithm 1 Constructing flows by flow-augmenting paths

for $(i, j) \in E$ **do**
  $c_{ij} = a_{ij}$; // Put all costs equal to the input cost
end for
for $t \in T$ **do**
  for $(i, j) \in E$ **do**
    $f_{ij} = 0$; // Put the flow equal to 0
  end for
  repeat
    Apply (a modified version of) Dijkstra's algorithm to find Π = a cheapest flow-augmenting path to t in the residual graph of G;
    if a path was found **then**
      for $(i, j) \in \Pi$ **do**
        $f_{ij} = 1 - f_{ij}$;
      end for
    end if
  until no path was found
  Define the flow as $B_t = \{(i, j) \in E : f_{ij} = 1\}$;
  for $(i, j) \in B_t$ **do**
    $c_{ij} = 0$; // Reusing (i, j) for another sink is free
  end for
end for
return $\{B_t : t \in T\}$

4 Conclusion

We have presented heuristics for flow graph selection for network coding in the case of unit link capacities. For instances where cyclic flow graphs are permitted, we have suggested a method based on linear programming. The other approach is based on a modified version of Dijkstra's algorithm, and is aimed for instances where the flow graph is required to be acyclic.

By comparison to lower bounds produced by linear relaxations of the flow selection problem, computational results published in [10, 11] indicate that the results produced by our methods are very close to the optimal solution.

References

[1] L. R. Ford, D. R. Fulkerson, "Maximal Flow through a Network", *Canadian Journal of Mathematics*, 8 (1956), pp.399–404. 3

[2] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", *IEEE Transactions on Information Theory,* Vol. 46, April 2000, pp. 1204-1216. 1

[3] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, Shi Jun and B. Leong, "A Random Linear Network Coding Approach to Multicast", *IEEE Transactions on Information Theory,* Vol.52, October 2006, pp.4413-4430. 1

[4] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain and L.M.G.M. Tolhuizen,"Polynomial Time Algorithms for Multicast Network Code Construction",

IEEE Transactions on Information Theory, Vol. 51, June 2005, pp. 1973-1982. 1, 2.1, 3

[5] Á. Barbero and Ø. Ytrehus, "Cycle-logical Treatment of 'Cyclopathic' Networks", *IEEE Transactions on Information Theory,* Vol. 52, June 2006, pp. 2795-2805. 1

[6] Á. Barbero and Ø. Ytrehus, "Knotworking", *Proceedings of ITA 2006,*, San Diego, February 2006 (electronic publication). 1

[7] T. Cui and T. Ho, "Minimum Cost Integral Network Coding", *Proceedings of IEEE Symposium on Information Theory, 2007* 1, 1.1, 3, 3.1

[8] K. H. Rosen (Ed.), "Handbook of Discrete and Combinatorial Mathematics", *CRC Press*, 1999. 3.2

[9] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-Cost Multicast over Coded Packet Networks", *IEEE Transactions on Information Theory*, Vol. 52, Issue 6, June 2006, pp. 2608– 2623. 3, 3.1.1

[10] M. Ravanbakhsh, Á. I. Barbero, D. Haugland and Ø. Ytrehus,"Power savings of cyclic network coding for multicast on wireless networks", *IEEE Information Theory Workshop*(ITW), Cairo, Egypt, 2010. 4

[11] M. Ravanbakhsh, *Towards Optimal Data Transmission by Network Coding*, PhD Dissertation, University of Bergen, Norway, 2009. 4