View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by eScholarShare at Drake University

Classifying Networks for Network Coding

William Janssen and Eric D. Manley
Department of Mathmatics and Computer Science
Drake University
Des Moines, IA 50311
{william.janssen,eric.manley}@drake.edu

Abstract

Network coding is a relatively recent development in the realm of maximizing information transfer in communications and computer networks. Traditional networks operate by simply storing and forwarding data along. Network coding, however, allows intermediate network nodes to combine data using arithmetic operations. In many instances, this can lead to more efficient use of network resources. Since there is a significant throughput input in some networks, some studies have been done on what kinds of networks will benefit from coding. A coding advantage is defined as a situation where a network coded graph has a lower cost to send given information per unit time session than the same un-coded graph. It has been proven that for two simple single-sender-single-receiver communication sessions that a graph must have one of two special graph-theoretic structures called the *butterfly* and *grail* in order to yield a coding advantage. We decided to focus our efforts on a different traffic scenario: a multicast session with a single sender and multiple receivers. Through our research we proved that a multicast-version of the butterfly network structure is needed within a single session multicast with two sinks and one source in order to gain a coding advantage. We also performed a simulation-based study in order to study the structures of multicast sessions with a larger number of receivers. The study involved the random generation of networks using several graph generation techniques. We also considered a variety of different edge-weighting constraints. Given a particular graph with set edge weights, the coding advantage problem was modeled as a linear program and run through the simulator to determine if a coding advantage was gained. Based on visual inspection of these results, it appears that variations of the multicast butterfly are ultimately the dominant structure allowing for a coding advantage. We also found that many types of random networks only very rarely resulted in a coding advantage. Only the graphs generated using the rectangular grid method showed a coding advantage, with a coding advantage percentage of 0.005% for 4 sinks in a 30 node network, with the coding advantage percentage going up as the number of sinks within the network increased.

1 Introduction

Network coding is a relatively recent development in the realm of maximizing information transfer in communications and computer networks. It can allow for more efficient uses of network resources by combining data instead of the traditional method of storing a copy of the data and forwarding it along. Since there is a significant throughput input in some networks, some studies have been done on what kinds of networks will benefit from coding. It has been proven that for two simple unicast sessions that a graph must have either a *butterfly* or *grail* structure to gain a coding advantage. [3] We prove a similar result for a single multicast network with two sinks and a single source.

2 Network Coding

Computer networks can be represented by a mathematical model known as a graph. Since graph theory has already been extensively researched, many results can be used to help solve problems in network theory. Every graph contains a set of nodes and a set of connections, either directed or undirected, known as edges, between those nodes. Each edge is assigned a maximum capacity of information it can carry. A computer network simply has computers or servers as the nodes and the set of connections. The source node can be described as the *sender* of information, and the set of sink nodes can be described as the *receivers* of that information. [5] There are three types of single source networks, categorized by the number of sinks in the network. A unicast has a single sink, a multicast has multiple sinks, and in a broadcast network every node is a sink [4].

In traditional networks without network coding, units of information are simply copied and forwarded along toward their destination. Using network coding data can be combined, allowing for increased throughput and protection against network failure. Previous research has shown that for two unicast sessions in the same simple network, there is a network coding advantage if the graph contains a *butterfly* or *grail* structure. [3] Our research built upon these findings, as we looked into the possibilities for a single multicast session.

3 Experimental Approach

A linear program is a mathematical model for determining the most beneficial outcome. A linear objective function that describes the problem is created, subject to linear equality or inequality restraints that represent the conditions. Using the given constraints, the objective function is either minimized or maximized depending on the problem. This algorithm was very useful because the network advantage problem can be modeled as a linear program. Sage, an open-source mathematics software package, was used to create the experimental programs. Sage was chosen for three reasons: it has an intuitive Python interface as well as an included Integer Linear Program Solver and useful network studying package networkX.

The main goal for the program was finding a minimum-cost multicast with network coding. We modeled the program as an integer linear program for finding variations of Steiner trees in graphs. Once this program was built and running, it was important to be able to represent

the results both empirically and visually. This module was then taken and put onto a server to run large scale simulations on where a coding advantage was gained within a graph. A program was written to run the Steiner tree solver twice for each input graph. The first time through the program, the variables were fixed as integers (representing the least-cost multicast without network coding), while the second time through the variables were allowed to be real numbers (representing the least-cost multicast with network coding) [2]. If the real solution cost was less than the integer solution cost, the network gained a coding advantage and the graph data was recorded to be analyzed.

We studied several classes of random networks. A first attempt was made using the built-in random graph generators provided with SAGE and Networkx, but this provided no instances of networks with a coding advantage. After some research into other papers, a promising technique was found called rectangular grid generation. In this method, first a rectangular grid is constructed and then nodes are randomly assigned to points on the grid. Then links are made between two nodes based on the probablility equation

$$P = \beta * e^{-d(x,y)/L*\alpha}$$

Where L is the maximum distance between two points in the whole grid, $d(x, y)$ is the manhattan distance between points x and y, α is the ratio of longer edges to shorter edges, and β is the node degree. [1]

4 Simulation Results

Using the rectangular grid graph generator, thousands of random graphs were created and run through the simulator on a dedicated SAGE server. First, the graph generator was set up to generate random graphs with each edge connection being held at a constant weight of 1.0. The number of sinks in the graph was increased each simulation, ranging from 3 up to 30. Every sink number simulation was ran one thousand times. Results were tracked as the number of graphs that gained a coding advantage, the number of graphs that contained unfeasible solutions, the degree of coding advantage gained, and the actual graphs themselves. Figure 1 shows our results from this experiment. Our initial assumption that a very small percentage of graphs would gain a coding advantage, and this was backed by our results.

Our next experiment was to use the same rectangular grid graph generator to generate random graphs with a random edge weight. Each edge connection was given a random weight, ranging from 1.0 to 10.0. Again, one thousand simulations were run for a given number of sinks, ranging from 3 to 30. The results were similar to the experiment when the edge weight was held constant at 1.0. Both had a very small percentage of graphs where network coding gained an advantage, and both had a spike around 8-9 sinks in a graph, as seen in Figure 2.

The next step was visualizing the graphs that gained a coding advantage. Graphs were redrawn from node and edge data, once with network coding and once without. A third graph was drawn comparing the two to find the differences. Once a collection of graphs

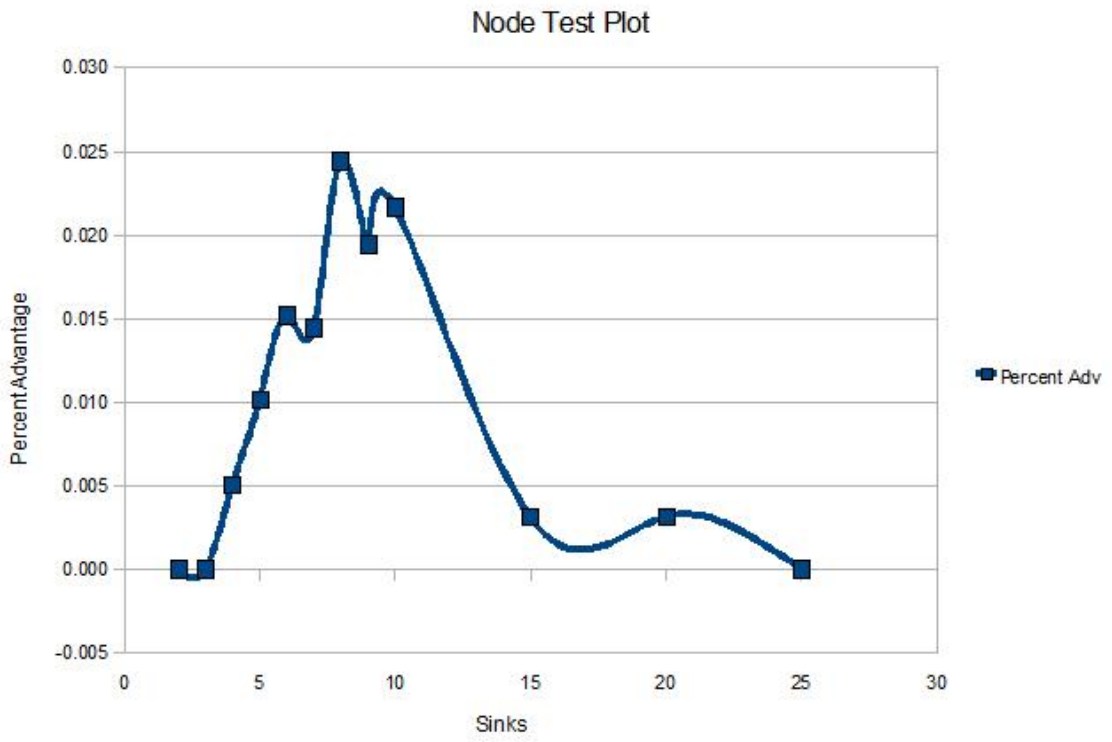

Figure 1: A plot of the percent coding advantage versus the number of sinks in the graph, with edge weight held constant at 1

large enough to start analyzing was gathered, each graph was broken down into Steiner trees to compare the paths with and without network coding present. Then, the smallest possible sub-graph of each that gained an advantage from network coding was picked out and compared to the rest to try and find a common structure. This gave a starting point to work towards in coding advantage proof.

5 Coding Advantage in a Single Multicast

This section considers the coding advantage in networks containing a single source and two sinks. For simplicity, we assume all edges have unit capacity and cannot be allocated fractionally. Let the source node be labeled s and the sink nodes u and v. A multicast transmission rate of k can be supported without network coding if there exists k edge-disjoint directed Steiner trees (i.e. a subgraph containing s to u and v). With network coding, a multicast transmission rate of k can be supported if there exists k edge-disjoint directed paths between s and u as well as s and v [5].
A graph has the *butterfly structure* if it contains directed paths from source s to the sinks u and v as shown in Figure 3.
Our main result in this section is the following theorem:

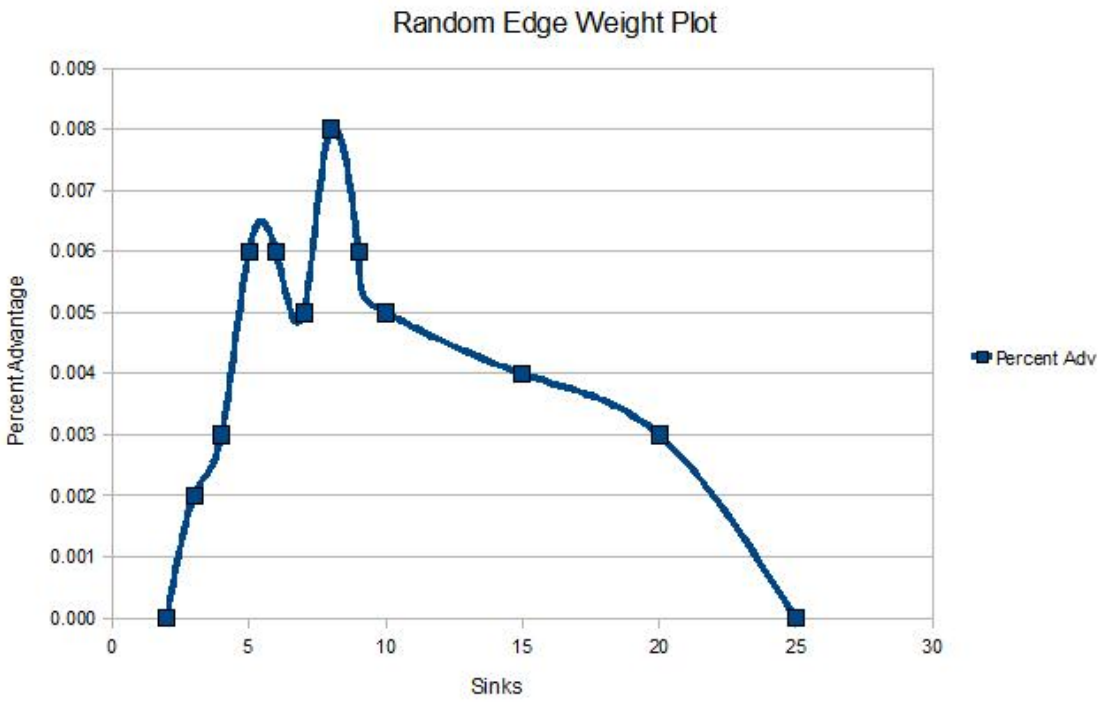

Figure 2: A plot of the percent coding advantage versus the number of sinks in the graph, with each edge given a random weight

Theorem 1. *If an acyclic network supports only a single unit of multicast without network coding, then it only gains a coding advantage if it contains the butterfly structure.*

Proof. We say that two paths intersect if they share an edge. When we say one path leaves another path, it is leaving from the most downstream shared edge. Since the network has a maximum non-coded multicast rate of one, at most one Steiner tree can be packed in the graph. If the network gains a coding advantage, then there exists at least two edge-disjoint paths between s and u and two edge-disjoint paths between s and v. These four paths are denoted as $s - u_1$, $s - u_2$, $s - v_1$, and $s - v_2$. We now consider how these paths might intersect each other.

First, assume that the two $s - v$ paths are disjoint from the two $s - u$ paths. In this case, two edge-disjoint Steiner trees can be formed: Paths $s - u_1$ + $s - v_1$ form one Steiner tree, and paths $s - u_2$ + $s - v_2$ form the other. Therefore two Steiner trees can be packed into the graph and there is a contradiction.

Now assume to contradiction one $s-v$ path is independent of both $s-u$ paths but intersects one of the $s - u$ paths at some point y. It is allowed to leave this path and rejoin at another node, or leave and intersect with the $s - u_2$ path any multiple number of times. It leaves either $s - u$ path at the furthest downstream node x, and from there travels to sink v. In this instance, two edge independent Steiner trees can be formed as well. The paths $s - u_1$ + $x - v$ form one Steiner tree, while the direct paths $s - u_2$ + $s - v$ form a second Steiner

tree. Therefore two Steiner trees can be packed into the graph and there is a contradiction.

From these two cases, we draw the conclusion that the two $s - v$ paths must intersect the $s - u$ paths in some fashion. By symmetry, we can conclude that both $s - u$ paths must intersect the $s - v$ paths in some fashion. Next we look at the case when each $s - v$ path intersects at least one of the $s - u$ paths. Fist assume both $s - v$ paths leave from a different $s - u$ path at the furthest downstream points x and y on each $s - u$ path to sink v. In this instance, the paths $s - x + x - u + x - v$ form one Steiner tree, and the paths $s - y + y - u + y - v$ form another edge independent Steiner tree. Therefore two Steiner trees can be packed into the graph, and there is a contradiction.

At this point we have drawn three conclusions:

1. Both $s - u$ paths must be intersected at least once by an $s - v$ path
2. Both $s - v$ paths intersect some $s - u$ path
3. Both $s - v$ paths last intersected edge must be on a common $s - u$ path. WLOG, call this path $s - u_2$

Thus, the graph must have a node w on path $s - u_1$ that connects with some node x on path $s - u_2$, where path $w - x$ is the most downstream such path which is part of either $s - v_1$ or $s - v_2$. $w - x$ is either upstream of point y, inbetween points y and z, or downstream of point z (where point z is the point where $s - v_2$ leaves $s - u_2$ for sink v). Thus, $s - v_1$ contains y and $s - v_2$ contains z. Taking the case where path x is upstream of y first, paths $s - v_1$ and $s - v_2$ cannot share path $x - y$ since they must be independent of each other. There must be another path for $s - v_2$ to get to point z.

Case 1: If $w - x$ is part of $s - v_1$, and therefore contains path $y - v$, in order to maintain assumptions there must be a way for $s - v_2$ to *jump* around the portions of path $x - y$ that are contained on $s - v_1$. For some points r and t on $s - u_2$, where r is up to but not including node x and t is from but not including node y up to but not including node z, there must be a path $r - t$ such that path $s - v_2$ includes $r - t$ and is independent of $s - v_1$. In this instance, paths $s - w + w - u + w - x + x - y$ (the portion of $s - v_1$ starting at x and ending at y) + $y - v$ form a Steiner tree, and paths $s - r + r - t + t - z + z - v + z - u$ form another Steiner tree. Therefore there are two edge independent Steiner trees, and two Steiner trees can be packed into the graph which is a contradiction.

Case 2: If $w - x$ is a part of $s - v_2$,and therefore contains path $z - v$, there must be a path that leaves before node y and after node x that *jumps over* node y and re-enters $s - u_2$ before node z. The path also may intersect with the $s - u$ paths along the way so long as it remains independent of $s - v_1$. Let s', t, t', and r be nodes on $s - u_2$, where s' is up to but not including point x, t is downstream of x up to but not including point t', t' is downstream of t up to but not including y, and r is downstream of y up to but not including z. There must be at least path $s' - t'$ and $t - r$ to keep all assumptions correct. The pattern of each $s - v$ path *jumping over* each other to keep $s - v_1$ and $s - v_2$ independent can happen multiple times as long as the final *jump* is the $s - v_2$ path going around point y and reaching point r. In this case, only one independent Steiner tree can be packed into the

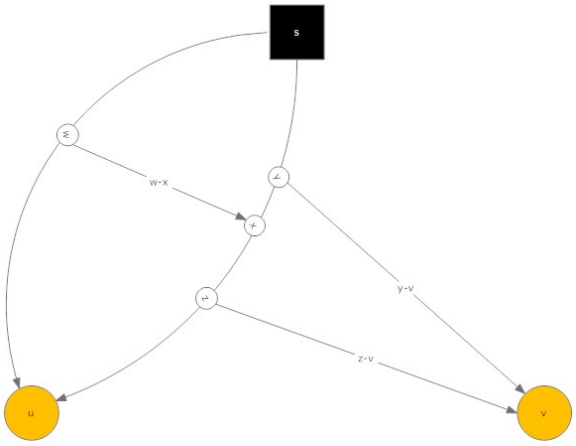

Figure 3: The Butterfly Network

graph per time unit. The resulting graph has the same basic structure as a butterfly, and can be classified as a butterfly because the added edges do not change the characteristics of the graph.

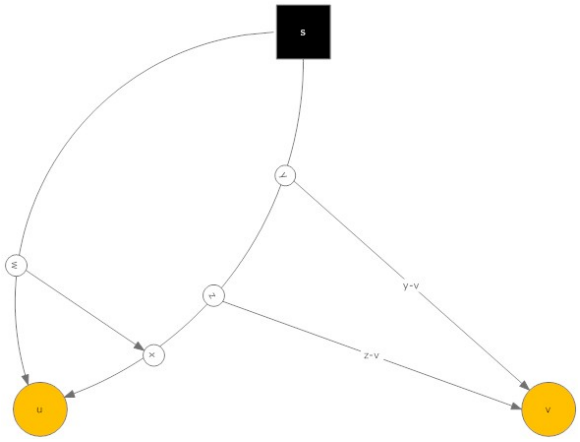

Figure 4: Case 3 Below

Now if we look at the case where path $w - x$ is downstream of point z in the graph, see Figure 4, any graph that has such a characteristic is cyclic. This is because x is downstream of both $s-v_1$ and $s-v_2$. No matter which of those paths contains $w-x$, the other will have *jump* over a portion of $s - u_2$ that contains x. This jump, and the fact that the $s - v$ path that does include $w - x$ must travel upstream, creates a cycle within the graph. Therefore, there is a contradiction for all graphs in this case.

The final instance of the graph is when path $w - x$ connects both $s - u$ paths inbetween points y and z, as in , as in Figure 3. In this case, all assumptions are true and only one Steiner tree can be packed into the graph per time unit. The shape of this graph is the butterfly.

Therefore, the only graph topology that supports a network coding advantage in a single multicast with two sinks and one source is the butterfly graph. QED

□

References

[1] LI, Z., LI, B., AND LAU, L. C. On achieving maximum multicast throughput in undirected networks. *IEEE/ACM Trans. Netw. 14*, SI (2006), 2467–2485.

[2] MANLEY, E. D. *Network Coding for WDM All-Optical Networks*. PhD thesis, University of Nebraska - Lincoln, 2009.

[3] WANG, C.-C., AND SHROFF, N. B. Beyond the butterfly - a graph-theoretic characterization of the feasibility of network coding with two simple unicast sessions. In *IEEE International Symposium on Information Theory (ISIT)* (June 2007), pp. 121–125.

[4] WEST, D. B. Introduction to graph theory, 2nd edition, 2001.

[5] YEUNG, R. W., LI, S.-Y. R., CAI, N., AND ZHANG, Z. *Network Coding Theory*. now Publishers Inc., Hanover, Massachusetts, USA, 2006.