Channel Management Strategies for Profitable Lightning Nodes

Sassan Hashemi

February 2022

Contents

Abstract

Bitcoin, an emerging digital asset, has limited transaction throughput (Nakamoto 2008). The leading solution to expand transaction throughput is currently the Lightning Network, a graph of payment channels that allow for transactions to clear in a few seconds at almost no cost. Routing nodes in this network collect a fee for forwarding payments, creating a profit incentive for participation (Poon, Dryja 2016). Our work looks to find different methods of creating new edges in this graph to maximize the profit generated by a participant. We explore different centrality measures (Freeman 1991, Freeman 1977, Page 1999) that reflect the frequency of nodes routing transactions, as well as different methods of randomly sampling transactions to measure fees collected. We use this array of metrics to measure the effectiveness of different edge addition algorithms subject to cost constraints. We found that random sampling measures were more correlated with each other, while centrality measures were more correlated with each other. Additionally, we found that while there is no conclusive evidence on which channel management strategy is best, our results suggest the algorithm introduced by Parotsidis (Parotsidis et al. 2016) may be generating the most fee revenue. Further experiments are needed to discover which channel management strategy maximizes revenue.

1 Introduction

1.1 Bitcoin

Bitcoin is an emergent digital asset that can be sent over a peer to peer network without the need for a trusted third party (Nakamoto 2008). Transactions are verified using public key cryptography on a distributed network of computers, each maintaining a copy of a public ledger. This distributed ledger, along with proof of work, prevents participants from double spending bitcoins and allows anybody to audit every transaction as well as the supply of bitcoins. The ledger enables transactions without an intermediary with the trade off of extreme redundancy.

1.2 Lightning

One constraint of using a redundant distributed ledger to record transactions is a limit to the number of transactions per second. A promising solution to increase transaction throughput involves a graph of payment channels (Poon, Dryja 2016). By keeping track of transactions separately, only publishing the aggregate to the ledger, throughput can be increased by batching transactions together. Consider Alice and Bob, two participants that transact often. Instead of cryptographically signing and broadcasting every transaction for the distributed ledger, they can instead keep track of the total each owes the other while signing promises to pay. This enables either of them to settle unilaterally at any time. Any pair of participants can similarly set up a payment channel, creating a graph where participants are nodes and payment channels are edges.

Using hashed timelocks and secret sharing (Poon, Dryja 2016), transactions can be routed through this network, given there is a path with sufficient liquidity to enable the payment. To incentivize users to participate in forwarding transactions, each intermediary node is paid a small fee to route transactions.

1.3 Fee estimation experiment

The first part of our work looks for methods of measuring fees collected by each node in this graph. Since transactions are onion routed (Poon, Dryja 2016), it becomes difficult to gain visibility into each transaction, the routes taken, and the fees collected. For this reason, our methods of estimating fees collected include centrality measures and random sampling of transactions from different statistical distributions. We believe that no single method is perfect at estimating fees, so using a wide array of different methods provides more insight into the fees collected by each node.

1.4 Channel management experiment

Our work builds on measuring fees to compare strategies for a new node joining the graph wanting to maximize fees collected given a set of constraints. We begin by developing and testing methods to estimate fees collected by each node based on graph topology. We then test three different algorithms for connecting a new node to the graph, using the number of edges as a constraint. Finally, we compare the results of three edge addition methods based on our various metrics for estimating fees.

1.5 Lessons from experiments

We learned that centrality measures were all more correlated to each other, while random sampling results were correlated to each other. Surprisingly, Pagerank (Page 1999), rather than betweenness centrality, is the centrality measure containing the most overlap with random sampling results. From this, we learned that if random sampling is the most accurate way to measure fees collected, centrality measures are not as useful.

We also found that for the three edge addition algorithms we used, there was an inverse correlation between the estimated fees collected through random sampling and centrality scores. This result may indicate that certain shortest paths are more important than others, as some nodes transact more than others. The greedy betweenness centrality maximization algorithm proposed by Bergamini (Bergamini et al. 2017) performed best on betweenness centrality, while the algorithm introduced by Parotsidis (Parotsidis et al. 2016) performed best on random sampling.

1.6 Importance for computer science

These findings may provide further insight into interactions within pseudonymous networks or payments networks. By using a similar set of centrality or fee estimation measures, other networks such as Twitter's social graph or Venmo public transactions can be analyzed to uncover how information flows pass through specific nodes. These results can be used to determine critical nodes in these graphs as well as what connections enable one to become a central node in the graph.

2 Background Information

2.1 Lightning network

In order to ensure the Bitcoin ledger remains affordable to audit as it grows, there are limits to the total memory used by transactions that can be included in it (Bitcoin 2009). A fee market develops where the transactions who bid the highest fee are appended to this ledger, while other transactions must wait. This results in transactions often taking 10 or more minutes to confirm with fees sometimes northward of one USD. While this may be sufficient for large transactions or international settlements, it becomes infeasible for smaller and more frequent payments. The naive solution of increasing the transaction throughput results in significant increases in the hardware and bandwidth costs of auditing the ledger, reducing the number of people enforcing the integrity of the ledger. This ultimately decreases the credibility of the claim Bitcoin makes to immutability and security.

The leading solution to the issue of long confirmation times and high fees is the Lightning Network. By digitally signing but not broadcasting transactions to the ledger, confirmation times can be reduced to seconds while a transaction fee can be distributed among thousands of transactions. The security of settlement comes from the ability of either party to settle to the ledger whenever they wish. Essentially, you have a fully collateralized promissory note that is always unilaterally redeemable. As more people create payment channels with each other, a graph develops with users as nodes and payment channels as edges. Payments can be routed through a path on the graph, with each intermediate node collecting a fee along the way (Poon, Dryja 2016). This creates an incentive for nodes to provide sufficient liquidity where demand for transactions exists.

Since payments are onion routed through the graph, only a subset of transaction information is publicly available. Each node can broadcast their payment channels and liquidity, the fees they charge, and other metadata if they choose to. They do not broadcast transactions that they have routed or fees that they have collected by routing transactions. Given the limited available information,

we are unable to fit our methods to ground truth data. Instead, we use centrality measures and random sampling of transactions to measure the performance of channel management strategies.

2.2 Centrality measures

The first centrality measure used in our fee estimation, betweenness centrality, measures each node's role in allowing information to flow through the graph (Freeman 1977). The betweenness centrality of a node is given by the following expression: $BC(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$ where σ_{st} is the number of shortest paths from s to t and $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through v (Freeman 1977). When relating this to the Lightning Network, we see that betweenness centrality is similar to the number of transactions that a node will route. Assuming every node has an equal chance of sending a transaction, and that the path with the least fees has sufficient liquidity, betweenness centrality would be equivalent to the number of transactions routed by each node given our assumptions in section 4.1.

Since betweenness centrality disregards liquidity or edge capacity, we use flow centrality as well. Flow centrality measures the total flow that passes through each node when measuring the maximum flow between every pair of nodes. The flow centrality of a node is given by the following expression: $FC(v) = \sum_{s \neq v \neq t} f_{st}(v)$ where $f_{st}(v)$ is the amount of flow that passes through v in the max flow from s to t (Freeman 1991). It serves as a measure of how central each node is for routing large payments through the Lightning Network graph. A larger flow centrality score means a node routes more large payments, while a smaller flow centrality score indicates a node routes fewer large payments.

Pagerank (Page 1999) measures the degree of importance a node has in a graph based on the eigenvectors of the adjacency matrix of the graph. The pagerank score of a node is the value of that node in the stationary distribution of the graph as a Markov chain (Page 1999). It was initially used by Google to identify important websites, but can also be applied to the Lightning Network graph to reveal important nodes as well. A node with a high pagerank score indicates other nodes with high pagerank scores have an edge pointing to this node. It is important to note that this centrality measure does not directly measure transaction counts or throughput, so it will probably be less accurate in measuring fees collected. Regardless, it remains important to use multiple different measures to compensate for the possible failures of one method.

2.3 K-Cores of a graph

A k-core of a graph is the maximal subgraph where the degree of every node is at least k (Batagelj 2003). The k-core graph is derived from a graph by repeatedly removing nodes with degrees less than k until all nodes have degree

k. Since the Lightning Network graph is a directed graph, we are using the sum of the indegree and outdegree when deriving the k-core subgraph. We use an $O(m)$ algorithm for generating the k-cores graphs (Batagelj 2003).

3 Lightning Network Graph Topology

3.1 Liquidity and degree distributions

While collecting informtaion regarding graph topology, we used the Lightning Network graph from September 16, 2021. Using the describegraph command on the Lightning Network Daemon (lnd 2017) command line interface, we probed the graph to record all public node and channel information. Many nodes and edges in this graph are not reachable by probing the graph, so those were not inclued in our experiments. While the public graph originally had 15363 nodes and 66473 edges, we pruned it by removing nodes that had less than 50,000 satoshis ($25; a satoshi is a subunit of a bitcoin, where each bitcoin is composed of 100 million satoshis) of total liquidity to improve run time. We assumed that nodes with low liquidity are less likely to route transactions as the transaction may require more liquidity than these nodes can support. The resulting graph had 596 nodes and 12990 edges. As shown in the figures below, the pruning reduced the number of nodes with low liquidity and low degree, while leaving the remainder of the graph mostly the same.

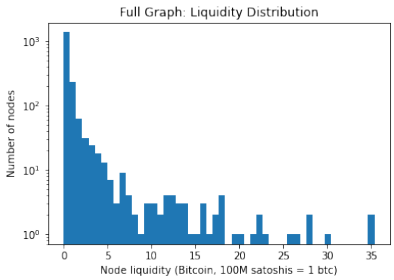

Figure 1: Full Graph: Liquidity Distribution

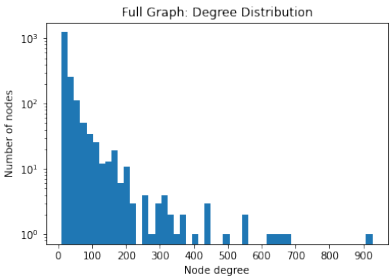

Figure 2: Full Graph: Degree Distribution

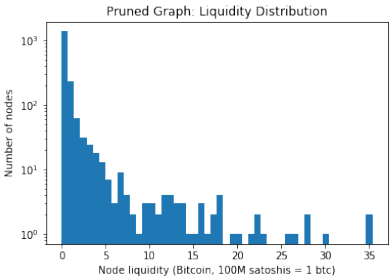

Figure 3: Pruned Graph: Liquidity Distribution

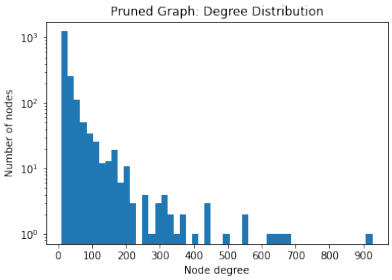

Figure 4: Pruned Graph: Degree Distribution

Based on the data in figures 1-4, we found that the distribution of liquidity among nodes was similar to the distribution of degree among nodes in both the original and pruned graphs. The large majority of nodes have a low degree and low liquidity, while very few nodes have many neighbors and large amounts of liquidity. We became curious if the nodes with high liquidity were the same as the nodes with high degree, as well as to what extent the degree and liquidity of these nodes were decreased while pruning. In order to determine whether the nodes were the same, we looked at the 50 nodes with the highest degree and the 50 nodes with the highest liquidity to measure ovelap. We found that in the original, pruned, and 10-cores graph, there were 26, 23, and 24 respective nodes overlapping. This shows about a 50% overlap between nodes with high degree and nodes with high liquidity. Additionally, the minimum, mean, and maximum liquidity of the top 10 nodes in the original graph was 8.83, 17.88, and 39.54 bitcoin respectively, while the minimum, mean, and maximum degree of the top 10 nodes in the original graph was 279, 597, and 2331 respectively. After pruning the graph, the respective values for liquidity became 7.06, 15.43, 34.10, while the respective values for node degrees became 157, 286, and 837. This information is included in figures 5 and 6. This indicates that while pruning the graph removed a significant number of edges from the highest degree nodes, it didn't significantly affect their liquidity. It seems that most of the liquidity for large nodes remains in edges with other large or mid sized nodes.

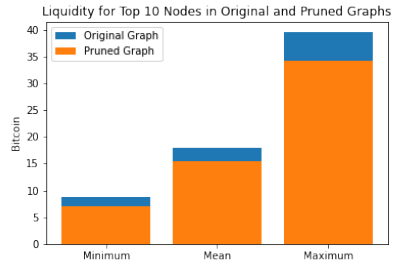

Figure 5: Liquidity for Top 10 Nodes in Original and Pruned Graphs

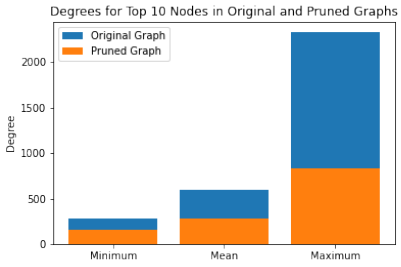

Figure 6: Degrees for Top 10 Nodes in Original and Pruned Graphs

Another interesting observation was that some nodes have over 20 bitcoin ($1,000,000) of liquidity, indicating that the Lightning Network has multiple wealthy participants or companies involved. Some of these nodes belong to companies providing services in the space including OpenNode, Bitrefill, and OkCoin (1ml 2022). There are other companies as well that maintain large Lightning Network nodes that are not publically broadcasted and thus not included in our experiment, including Block's Cashapp and Strike (1ml 2022). For these companies, maintaining many liquid channels that can connect to the rest of the graph becomes important when serving their customers who want to transact on the Lightning Network.

3.2 K-cores comparison

In order to determine the affect of pruning the Lightning Network graph, we compared the pruned graph to the k-cores graphs. Since k-cores graphs are well researched, we looked for similarities between metrics of our pruned graph and the k-cores graph. We began by taking the 1-core, 2-core, ..., 20-core graphs from our original graph. We then found the percentage of nodes that overlapped, as well as the intersection over union (IOU) of nodes and edges between the k-cores graphs and our pruned graph. The results are in figures 7, 8, 9, and table 1.

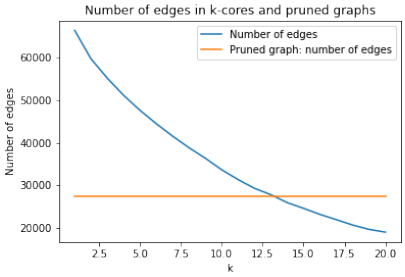

Figure 7: Number of Edges in 10-cores and Pruned Graphs

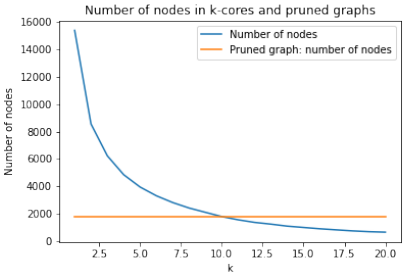

Figure 8: Number of Nodes in 10-cores and Pruned Graphs

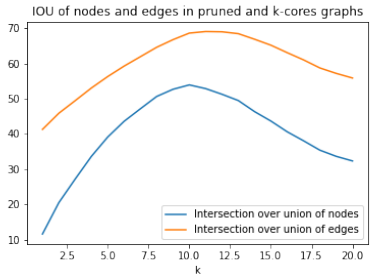

Figure 9: IOU of Nodes and Edges in Pruned and K-Cores Graphs

k	Nodes overlap (%)	Edges overlap (%)	Nodes IOU (%)	Edges IOU (%)
1	100	100	11.6	41.3
2	98.4	99.9	20.5	45.9
3	95.9	99.6	27.2	49.4
4	93.7	99.2	33.7	53.1
5	90.5	98.5	39.1	56.3
6	86.8	97.5	43.6	59.3
7	82.8	96.1	47.1	61.9
8	79.4	94.7	50.6	64.6
9	75.4	93.0	52.7	66.8
10	70.5	90.5	53.9	68.6
11	73.9	87.5	52.9	69.1
12	77.6	84.4	51.3	69.0
13	80.1	81.8	49.5	68.5
14	82.5	82.6	46.3	66.9
15	83.8	83.6	43.7	65.2
16	85.0	84.6	40.6	63.1
17	86.1	85.4	38.1	61.0
18	87.1	86.2	35.4	58.7
19	88.6	87.3	33.7	57.2
20	89.2	87.8	32.3	55.9

Table 1: K-Cores Comparison

The IOU of nodes and edges serves to determine the similarity of the two graphs. The IOU for nodes peaks at the 10-core graph, while the IOU of edges peaks at the 12-core graph. Due to the 10-core graph having an IOU for edges close to the peak, we determined that the 10-core graph is the most similar to our pruned graph of all the k-cores graphs. We then proceeded to measure the liquidity and degree distributions of the 10-core graph for comparison with our pruned graph. The results are in the figures 10 and 11.

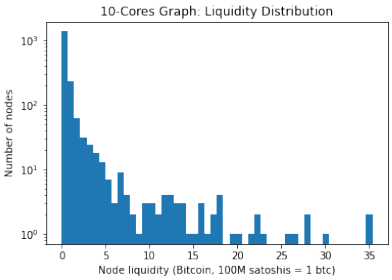

Figure 10: 10-Cores Graph: Liquidity Distribution

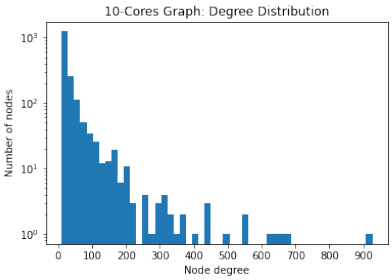

Figure 11: 10-Cores Graph: Degree Distribution

Looking at these tables, we found that the 10-cores graph has similar liquidity and degree distributions to our pruned graph. The degree and liquidity distributions, the IOU measures, and the overlap of nodes should serve to create somewhat of a link between the pruned graph and the 10-cores graph. While these graphs are not exactly the same, they share certain properties that may be useful for future researchers drawing insight from our work.

4 Experiment: Fee Estimation

4.1 Introduction, assumptions, and metrics

Given a graph, our goal is to construct a method that can determine if a node is collecting more fees than other nodes in the graph. This measure will be used in section 5 for determining the effectiveness of different channel management strategies. Due to the capital requirements for measuring real revenue on the Lightning Network, an estimate of fees collected derived through simulation serves as a proxy to filter out strategies without incurring significant costs.

In order to generate a method for routing fee estimation, we make a few assumptions about how transactions occur. While these assumptions may not perfectly reflect reality, they serve as a necessary basis on which to estimate fees.

1. When routing transactions, the sender will choose the path that invokes the lowest fee, disregarding other factors that determine the path used to route a transaction.

2. Payments will be routed along a single path, ignoring the possibility of multi path payments (MPP 2022).

3. Nodes with more outbound liquidity and channels are more likely to send transactions, while nodes with more inbound liquidity and channels are more likely to receive transactions.

The first assumption reflects the idea that individuals will act in in their best economic interest. While this does not perfectly model reality, it seems to be a reasonable assumption that can simplify our method of fee calculation without significantly contradicting reality. In reality, other factors such as node uptime, error frequency, and number of hops may be used when determining the route for a transaction. The second assumption simplifies the fee estimation process by making removing the case where a transaction takes multiple paths, with a different fee payout for intermediary nodes. The third assumption was made because many nodes with high liquidity and many channels act as custodial entities that provide lightning services to groups of individuals. It would thus make sense for these nodes to engage in more transactions than other nodes, as they aggregate the transactions of all their users.

While each fee estimation measure deviates from reality in different types of graphs, it is likely that any node collecting a significant amount of fee revenue will score highly on multiple metrics. This indicates that any node that scores highly on multiple different metrics may collect large fees, while those that score low across the board probably don't collect much in fees. Looking at success across multiple different metrics allows us to infer a node's relative revenue with more accuracy than any single measure.

4.2 Fee sampling methods

One method to estimate the fees collected by each node in the graph is by randomly sampling transactions and finding corresponding routes. Given a probability distribution over the set of nodes and payment amounts, we can sample a source and destination node, find the shortest (lowest fee) route with sufficient liquidity, and measure the fees collected by each node along the route. Repeating this sampling process will converge towards a more accurate measure of relative revenues generated by each node. Since the number of transactions per time frame is unknown and can vary, sampling transactions provides the relative revenues of each node rather than exact revenues over a specific time period.

When sampling nodes, we used a uniform distribution and a power law distribution with a parameter of 5. The power law distribution is taken over the set of nodes sorted by liquidity, degree, and randomly. For each distribution that we sampled from, we randomly generated 1500 transactions each moving 5000 satoshis ($2.50). We then measured the total fees collected by each node in the graph after routing these transactions along the shortest path.

We also used centrality measures to estimate fees collected by each node. While betweenness centrality is similar to an infinite number of random samples from a uniform distribution with a very small transaction amount, it doesn't factor in specifics of lightning transactions such as different transaction amounts

and distributions. We use betweenness centrality (Freeman 1977), flow centrality (Freeman 1991), and pagerank (Page 1999) to generate centrality scores for each node that are then ranked to determine relative fees collected.

4.3 Results and analysis

After using all of these metrics on the pruned lightning network graph, we compared the results to find overlap. We took the top 30 ranked nodes using each centrality measure and random sampling method and measured the overlap between each pair of measures. The results in table 3 use abbreviations from table 2 to save space.

Abbreviation	Measure
BC	betweenness centrality
FC	flow centrality
PR	pagerank
PD	power law distribution with nodes sorted by degree
PL	power law distribution with nodes sorted by liqudity
PR1	power law distribution with nodes sorted randomly (1)
PR2	power law distribution with nodes sorted randomly (2)
PR3	power law distribution with nodes sorted randomly (3)
UR	uniform distribution with nodes sorted randomly

Table 2: Abbreviations for Table 3

	BC	FC	PR	PD	PL	PR1	PR2	PR3	UR
BC	30								
FC	19	30							
PR	14	10	30						
PD	15	12	21	30					
PL	14	13	21	27	30				
PR1	16	13	20	24	22	30			
PR2	14	12	22	25	26	25	30		
PR3	18	12	20	23	23	23	24	30	
UR	16	10	18	22	22	22	22	22	30

Table 3: Fee Estimation Measures Overlap

Not surprisingly, we found that random sampling methods were more correlated with other random sampling methods, while centrality measures were more correlated with each other, regardless of the sampling distribution or centrality measure. Additionally, using a power law distribution with a parameter of 5 and nodes sorted by liquidity and randomly most closely mirrored the centrality measures, while uniform distribution random sampling was less correlated to centrality measures. Conversely, pagerank was most correlated to the random sampling methods. This differed from our expectation that betweenness centrality would most closely mimic random sampling methods as it counts the number of shortest paths, and our transaction routing takes the shortest paths.

With our goal of using fee estimation methods that cover a wide array of different measures, we used multiple uncorrelated measures. We chose betweenness centrality, random sampling from power distributions sorted by liquidity and randomly, and uniform random distributions. We chose more random sampling methods as we believe these more accurately measure the real fees collected. These metrics are used to rank our channel management algorithms in the following section.

5 Experiment: Channel Management

5.1 Introduction, constraints, and metrics

Our goal is to compare different methods of introducing new edges adjacent to a source node that maximize the fees collected by that source node given constraints. The source node is a new node that was introduced to the graph without any edges. It serves as the node belonging to the user whose goal is profit maximization. Since the ultimate aim is to maximize profits, and each new edge incurs a cost, we limit the number of edges that can be added. Our method involves adding edges between a source node and other nodes in the graph with 100,000 satoshis ($50) of liquidity in both directions, with each edge charging a fee of 1 satoshi per transaction. We perform these edge addiitons on the pruned graph. We add up to 16 edges while measuring the relative fee revenue using multiple different metrics from the previous section along the way. When deciding which edges to add, we use the three algorithms below.

5.2 Betweenness centrality maximization

This algorithm, introduced by Bergamini et al. (Bergamini 2017), seeks to add edges that maximize the betweenness centrality of a node in a graph. Each iteration, a new edge adjacent to the source node is added such that the resultant betweenness centrality of the source node is maximized. Pseudocode is included in algorithm 1.

Algorithm 1 Greedy Betweenness Centrality Maximization

Require: graph G, node s, integer $iters$
Require: $bc(G, n)$ be the betweenness centrality of n in G
Measure the betweenness centrality of all nodes in G
Create an edge between s and the node in G with the highest betweenness centrality
for $i : 1 \rightarrow iters$ **do**
    $betweenness\_centralities \leftarrow \{\}$
    for node n in G **do**
        Add edge between n and s
        $betweenness\_centralities[n] \leftarrow bc(G, s)$
        Remove edge between n and s
    Create an edge between s and the v where v has the largest value in $betweenness\_centralities$

5.3 Pagerank maximization

This algorithm, introduced by Larry Page (Page 1999), seeks to add edges that maximize the pagerank score of a node in the graph. Each iteration, a new edge adjacent to the source node is added such that the resultant pagerank score of the source node is maximized. Pseudocode is included in algorithm 2.

Algorithm 2 Greedy Pagerank Maximization

Require: graph G, node s, integer $iters$
Require: $pagerank(G, n)$ be the pagerank score of n in G
Measure the pagerank score of all nodes in G
Create an edge between s and the node in G with the highest pagerank score
for $i : 1 \rightarrow iters$ **do**
    $pr\_scores \leftarrow \{\}$
    for node n in G **do**
        Add edge between n and s
        $pr\_scores[n] \leftarrow pagerank(G, s)$
        Remove edge between n and s
    Create an edge between s and the v where v has the largest value in pr_scores

5.4 Parotsidis algorithm

This algorithm, introduced by Parotsidis (Parotsidis 2016), seeks to add edges that minimize the sum of shortest paths from the source node to all other nodes. Each iteration, an edge adjacent to the source node is added such that the sum of all shortest paths from the source node to all other nodes is minimized. Pseudocode is included in algorithm 3.

Algorithm 3 Greedy All Shortest Paths Minimization

Require: graph G, node s, integer $iters$
Require: $ssp(G, n)$ be the sum of shortest paths from n to all other nodes in G

```
Measure the betweenness centrality of all nodes in G
Create an edge between s and the node in G with the highest betweenness centrality
for i : 1 → iters do
    sums ← {}
    for node n in G do
        Add edge between n and s
        sums[n] ← ssp(G, s)
        Remove edge between n and s
    Create an edge between s and the v where v has the largest value in sums
```

5.5 Results and analysis

After running the three algorithms above while continually measuring betweenness centrality and randomly sampling fees collected, we produced figures 12-15.

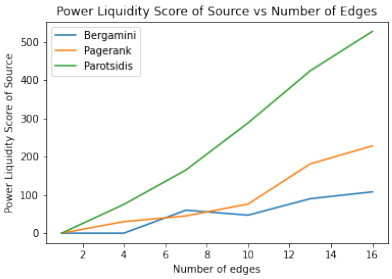

Figure 12: Scores for Power Distribution over Node Liquidities

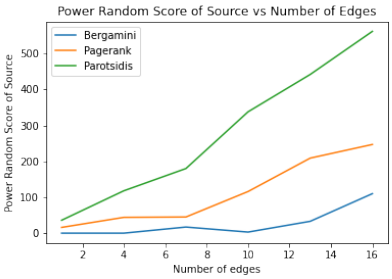

Figure 13: Scores for Power Distribution over Random Node Ordering

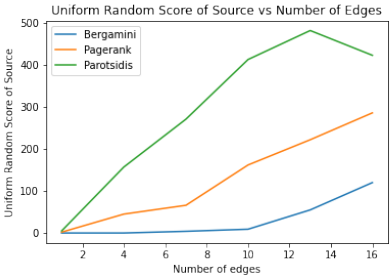

Figure 14: Scores for Uniform Distribution

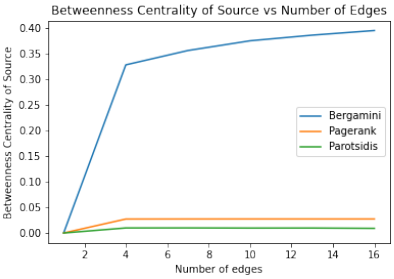

Figure 15: Scores for Betweenness Centrality

One insight across the board is that betweenness centrality and random sampling of fees are inversely correlated among different edge creating algorithms. The algorithm proposed by Bergamini had the highest betweenness centrality score and the lowest random sampling score, while the Paisitidis algorithm had the lowest betweenness centrality score and the highest random sampling score, and the Pagerank maximization algorithm was in the middle for both. While it is plausible that this is a coincidence due to the algorithms we chose or the structure of the graph, it is also possible that betweenness centrality and random sampling measure differing properties of the graph.

We also found that centrality measures increased rapidly during the first few edge additions, and slowly afterwards, while random sampling measures consistently increased. This indicates that only a few edges are required to have a high betweenness centrality score, while additional edges only provide minimal benefits. It also implies that the addition of new edges continues to improve random sampling measures without a significant reduction in marginal benefits.

Another finding is that most random sampling measures resulted in no fees collected after adding one edge, and a few had zero fees collected after adding 4 edges. When a node has one edge, it will not route any transactions unless it is the source or target node of the transaction. The random sampling measures that recorded a fee collected for a node with one edge did so by having that node be the start or end node of the randomly sampled transactions. The random sampling measures that show no fees collected with four edges may be due to the source node not being included in any of the random samples. While the likelihood of this reduces with the number of samples taken, it still remains possible.

We also looked at the distribution of nodes the source node created edges with. In Parotsidis's method of channel management (Parotsidis et al. 2016), the degrees of the nodes the source node connected to were widely distributed across the degree distribution for all nodes. The range of degrees for the nodes adjacent to the source node vary from 29 to 839, with the majority having less than 150 edges. When comparing this to the nodes with the 16 highest degrees (all have degree of at least 289), we found that for our most succesful channel management strategy, the source node does not simply connect to the nodes with the highest degree, but rather to nodes with varying degrees. These results are consistent with the liquidity of the nodes that formed new edges with the source node as well. Figures 16 and 17 includes the minimum, median, mean, and maximum degree and liquidity values of three different sets of nodes: the 16 nodes with the highest liquidity and degree, nodes that Parotsidis' algorithm connected to, and all the nodes in the graph.

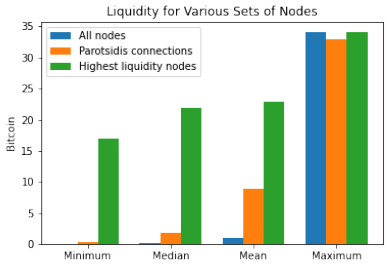

Figure 16: Liquidity for Nodes in Original, Pruned, and Top Graphs

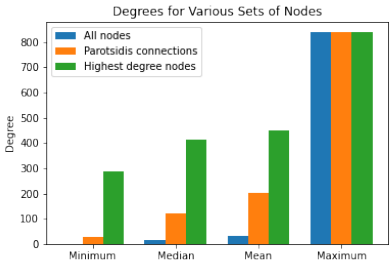

Figure 17: Degrees for Nodes in Original, Pruned, and Top Graphs

One possible explanation is that creating edges only with well connected nodes doesn't create any new shortest paths for any pairs of nodes. The source node must also create edges with nodes that are local hubs but not as well connected to the rest of the graph. We see 7 of the 16 edges with nodes that have fewer than 100 open channels. These nodes seem to be local hubs that are not directly connected to the rest of the Lightning Network. This indicates that connections between local hubs may be a void where demand for liquidity existed in September 2021. The types of nodes one should connect to in order to generate fee revenue may change over time as the topography of the Lightning Network graph develops.

6 Conclusion

6.1 Analysis of results

Our analysis of fee estimation metrics showed that random sampling measures were more correlated with other random sampling measures, while centrality measures were more correlated with each other, regardless of the sampling distribution or centrality measure. Surprisingly, betweenness centrality was not closely correlated to random sampling measures. While this could come from a small sample size of transactions, we suggest further research on the Lightning Network graph in different periods of time to investigate this result.

We also found that betweenness centrality and random sampling of fees may measure different properties of the Lightning Network graph. The algorithm proposed by Bergamini (Bergamini 2017) had the highest betweenness centrality score and the lowest random sampling score, while Parotsidis's algorithm (Parotsidis 2016) had the lowest betweenness centrality score and the highest random sampling score, and the Pagerank maximization algorithm was in the middle for both. While it is plausible that this is a coincidence due to the algorithms we chose or the topology of the graph, it is also possible that betweenness centrality and random sampling measure opposite or orthogonal metrics.

Due to the inconclusiveness of these results, further exploration using bitcoin to measure real fees is warranted. While our expectation of random sampling being a more accurate predictor of true fees indicates that the algorithm introduced by Parotsidis is most useful, a capital intensive experiment is warranted to confirm our results. Additionally, the experiment would be able to measure other factors used to determine routing paths that we ignored.

6.2 Importance of findings

These results may provide insight into pseudonymous social networks and how to model interactions between users. Pseudonymous actors still develop reputations and trust, but can have multiple dissociated identities. This may change the behavior that actors display when interacting with other members. Examples of pseudonymous networks include subgraphs of Twitter and Discord, where people can create multiple accounts not associated with real identities to interact with others.

6.3 Direction of future work

An area that is worth exploring is looking at the Lightning Network graph using a game theoretic framework. By identifying each node as a market actor looking to maximize their profits, it may be possible to determine if certain channel opening and fee setting strategies emerge as a Nash equilibrium, Schelling point, or game theoretic optimal strategies. While these results depend on the topology of the Lightning Network graph, they may also depend on the bitcoin denominated cost of capital and other advancements in the Bitcoin and Lightning ecosystems.

Recent advancements in the Lightning ecosystem allow for further exploration of strategies. For example, liquidity markets have emerged that allow people to purchase liquidity in channels from others for a fee. While we didn't analyze the cost structure of maintaining a Lightning node, liquidity markets could alter the number and size of channels that can be opened for a given cost. Additionally, the development of the bolt-12 (bolt-12) and lnurl (LNURL) standards allow for additional functionality that could affect how the Lightning Network is used. Bolt-12 allows for reusable invoices and payment promises that could alter the distribution and amount of transactions that occur. By introducing the ability to encode reusable invoices and offers to pay money, a QR code on a billboard or ATM can distribute or receive payments. Lnurl allows for url addresses to serve as Lightning addresses using API calls. This would allow for Lightning invoices to be built into hyperlinks, enabling different user experiences. These improvements significantly reduce the friction required to send and receive payments, potentially changing the nature of transactions that occur over the Lightning Network.

Discrete Log Contracts (DCI 2022) serve as another area of development on the Lightning Network, enabling conditional payments. Using discrete log contracts, two parties can create a contract that redistributes their funds depending on conditions that were met. The funds can be claimed by trusted parties digitally signing information, or by participants revealing the preimage of a hash. While enforcement of most contract law requires the threat of prison time from a military or police force, discrete log contracts on bitcoin are enforced by miners that include transactions in valid blocks.

Ultimately, the Bitcoin and Lightning protocol stack is nascent technology that has the potential to change the way value is transferred online. By creating economic incentives for actors to protect and improve the monetary network, Bitcoin allows for any two entities to transfer value over the internet without the need for a trusted third party. This paper investigates how to best capture those economic incentives to generate a profit, while uncovering interesting relationships between random sampling and centrality measures.

7 References

1. Nakamoto S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

2. Poon J., Dryja T. (2016) The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf

3. Freeman L., Borgatti S, White D. (1991) Centrality in valued graphs: A measure of betweenness based on network flow. Social Networks Volume 13, Issue 2, June 1991, Pages 141-154. https://www.sciencedirect.com/science/article/pii/037887339190017N

4. Freeman L. (1977) A set of measures of centrality based on betweenness. Sociometry 40(1), 35–41

5. Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry (1999) The PageRank Citation Ranking: Bringing Order to the Web. http://ilpubs.stanford.edu:8090/422/

6. Parotsidis N., Pitoura E., and Tsaparas P. (2016) Centrality-Aware Link Recommendations. WSDM '16: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (503-512). https://dl.acm.org/doi/abs/10.1145/2835776.2835818

7. Bergamini, E., Crescenzi, P., D'Angelo, G., Meyerhenke, H., Severini, L., and Velaj, Y., (2017) Improving the betweenness centrality of a node by adding links. ACM Journal of Experimental Algorithmics. https://arxiv.org/abs/1702.05284

8. Batagelj, Zaversnik (2003) An O(m) Algorithm for Cores Decomposition of Networks. Advances in Data Analysis and Classification, 2011. Volume 5, Number 2, 129-145. https://arxiv.org/abs/cs/0310049

9. Bitcoin. (2009) Bitcoin. https://github.com/bitcoin/bitcoin

10. LND (2017): https://github.com/lightningnetwork/lnd

11. 1ml (2022): https://1ml.com/node?order=capacity

12. Bolt-12 (2022): https://bolt12.org/

13. LNURL (2022): https://docs.lightning.engineering/the-lightning-network/lightning-overview/understanding-lightning-invoices

14. MPP (2022): https://docs.lightning.engineering/the-lightning-network/multihop-payments/multipath-payments-mpp

15. DCI (2022): https://dci.mit.edu/smart-contracts

8 Acknowledgements

I want to thank Professor John Gilbert for being such a supportive professor and advisor throughout this entire process. From three years ago when you helped me dabble in CS research with Cholesky factorization on sparse matrices, up until now where you helped me complete this masters project. I really appreciate how you never tried to push me towards any topic or in any direction, and instead helped me discover my own interests and supported my exploration. Thanks to you, I've been able to spend the last year exploring topics that peaked my interest, and dive deep into them. I really appreciate everything you've done to help me get to where I am right now. Thank you John.